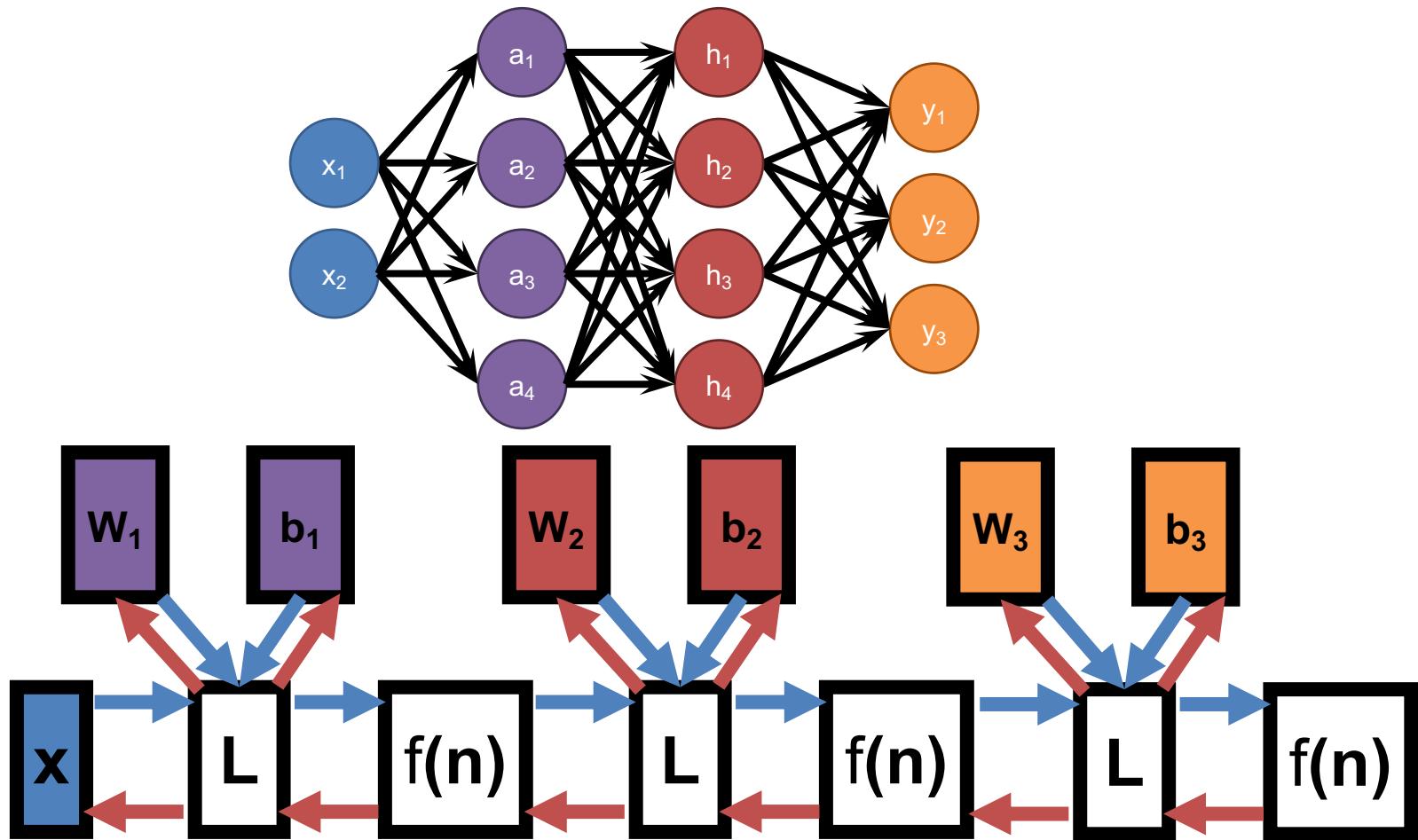


Convolutional Neural Nets II

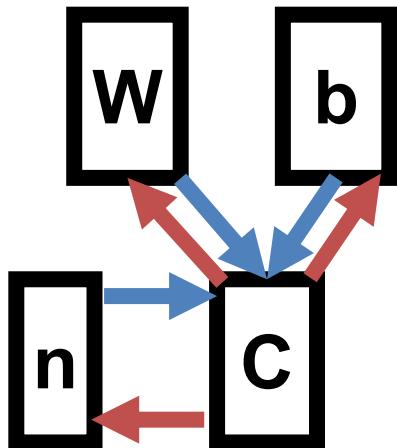
EECS 442 – Jeong Joon Park
Winter 2024, University of Michigan

Fully Connected Network



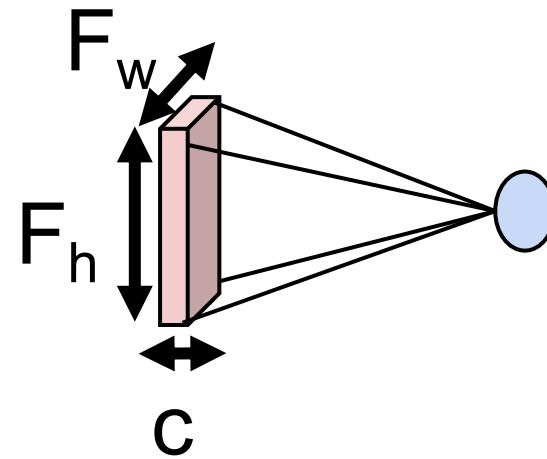
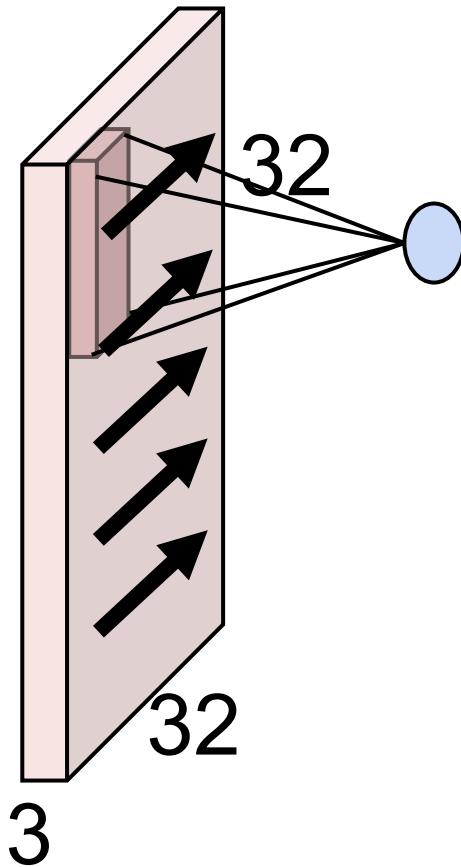
Convolutional Layer

New Block: 2D Convolution



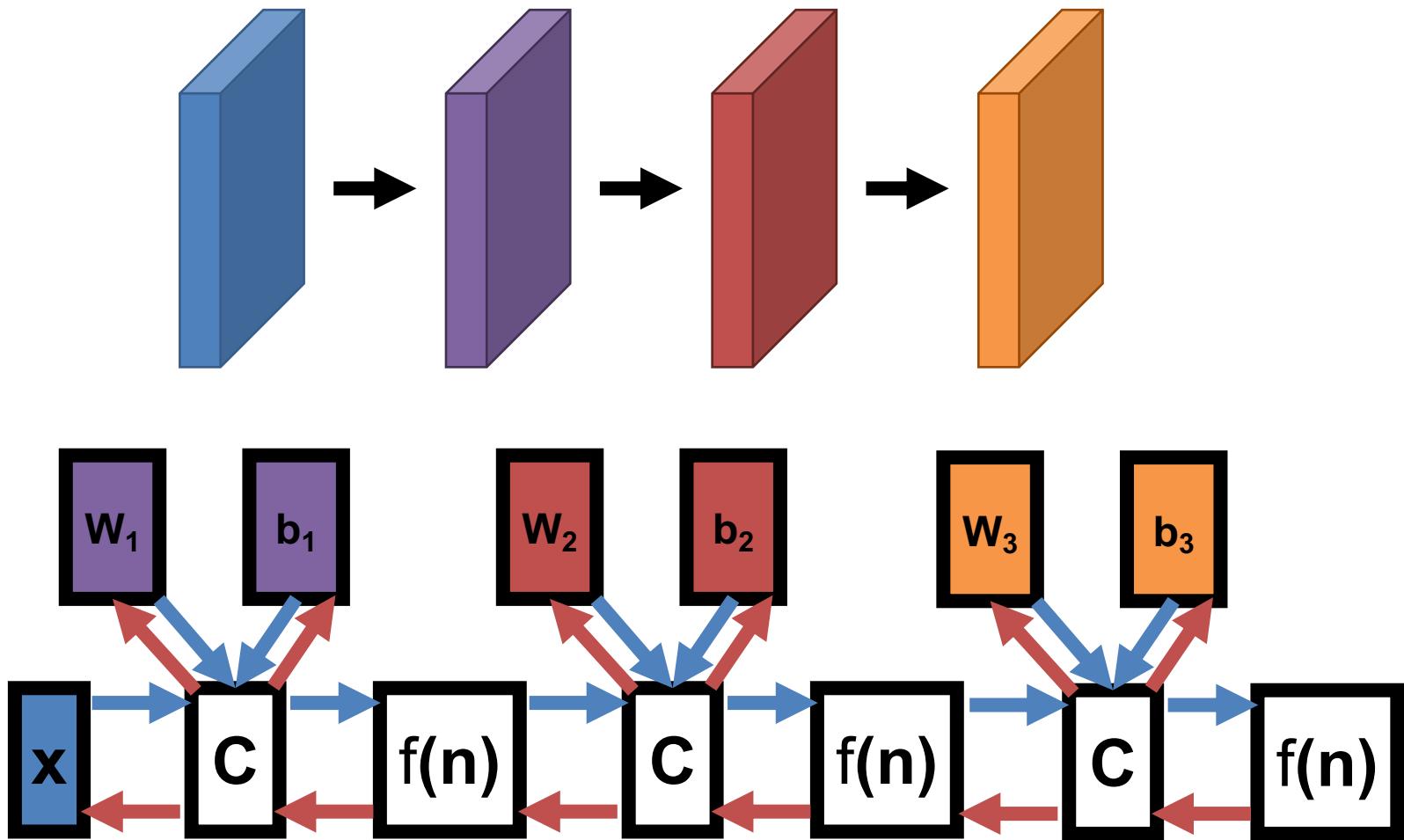
$$C(n) = n * F + b$$

Convolution Layer

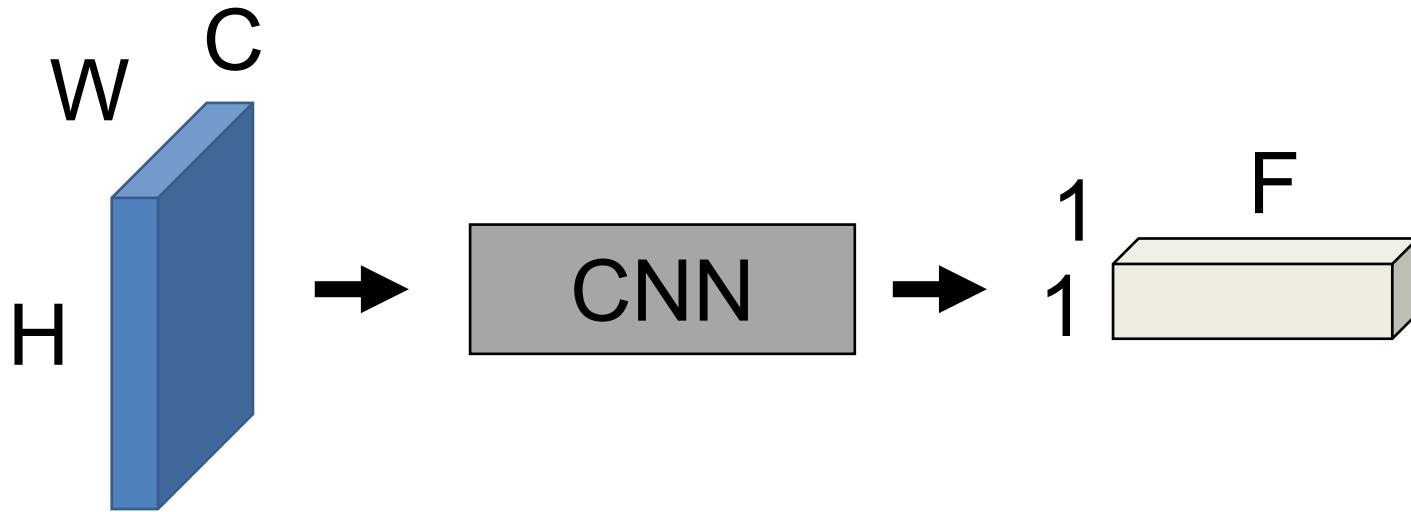


$$b + \sum_{i=1}^{F_h} \sum_{j=1}^{F_w} \sum_{k=1}^c F_{i,j,k} I_{y+i,x+j,k}$$

Convolutional Neural Network (CNN)



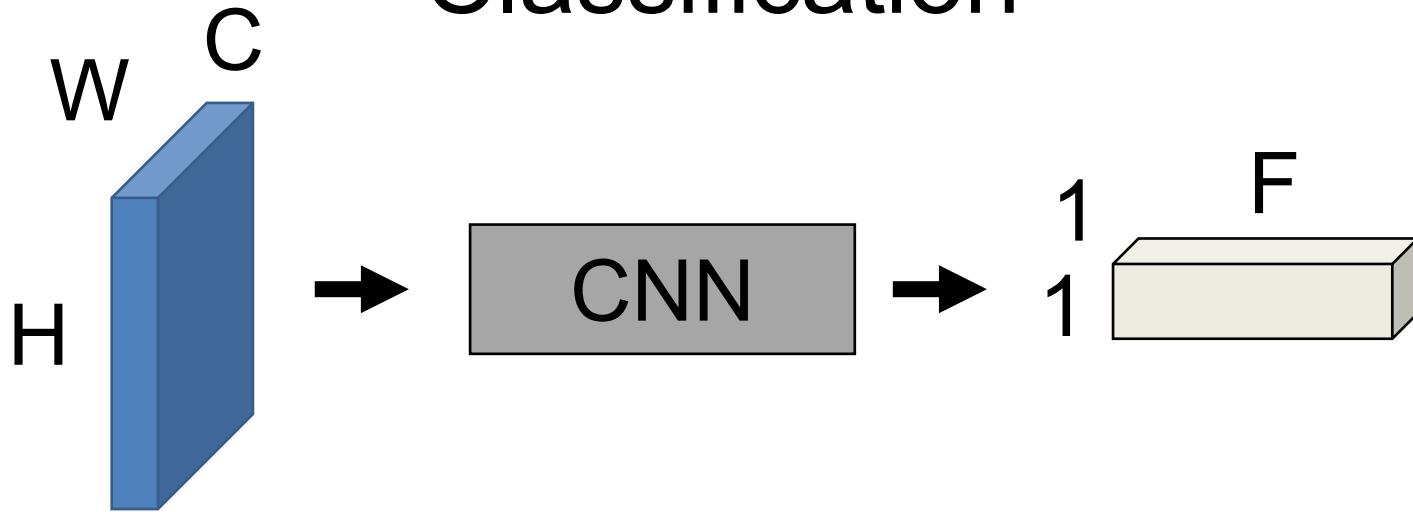
Today



Convert $H \times W$ image into a F -dimensional vector

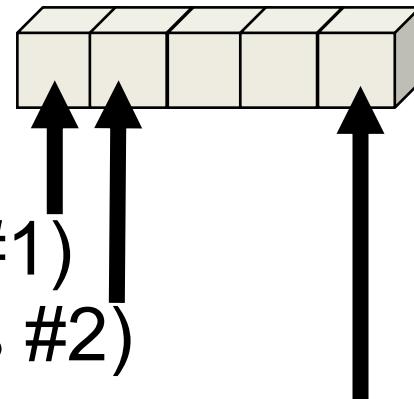
- What's the probability this image is a cat ($F=1$)
- Which of 1000 categories is this image? ($F=1000$)
- At what GPS coord was this image taken? ($F=2$)
- Identify the X,Y coordinates of 28 body joints of an image of a human ($F=56$)

Today's Running Example: Classification

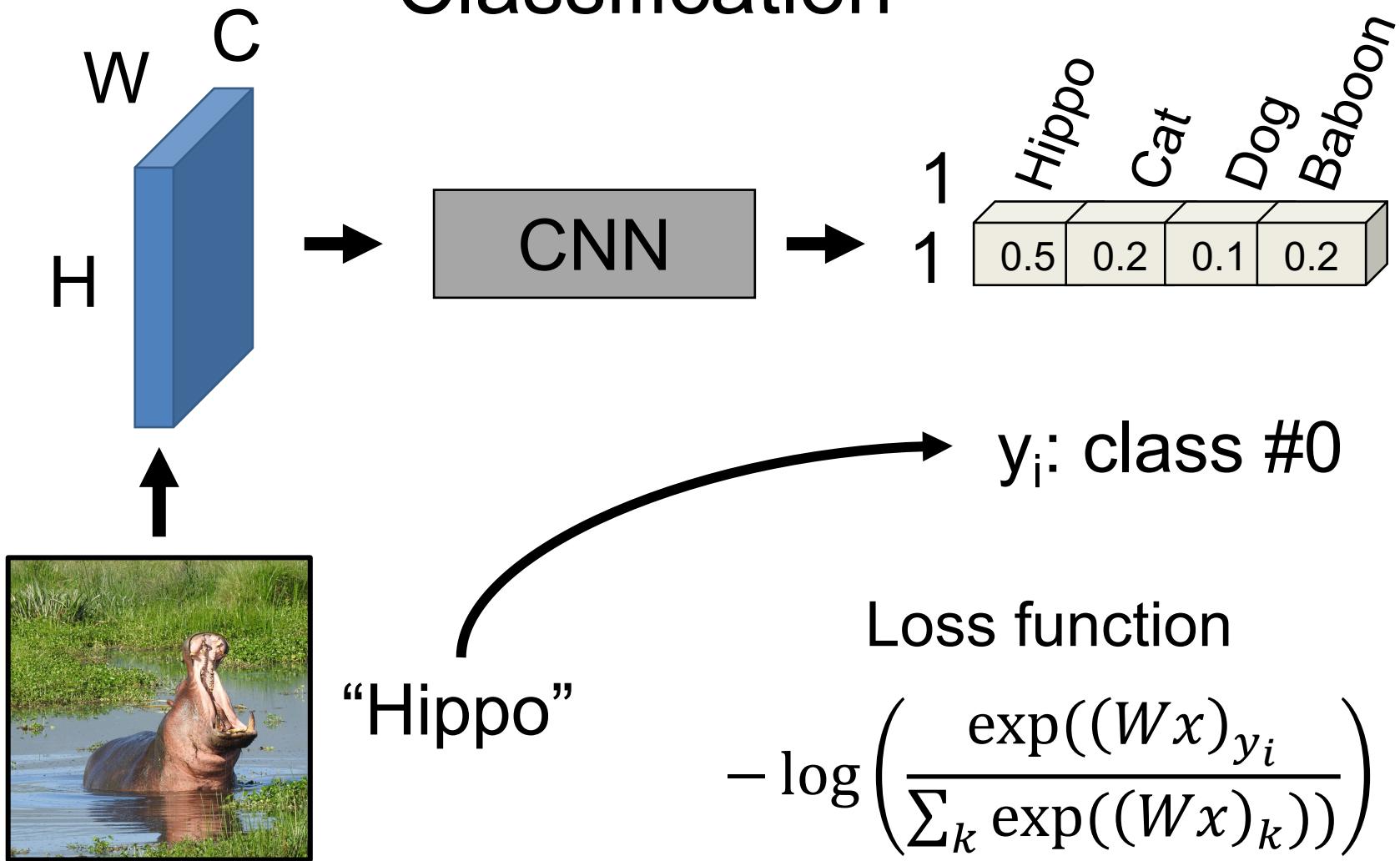


Running example:
image classification

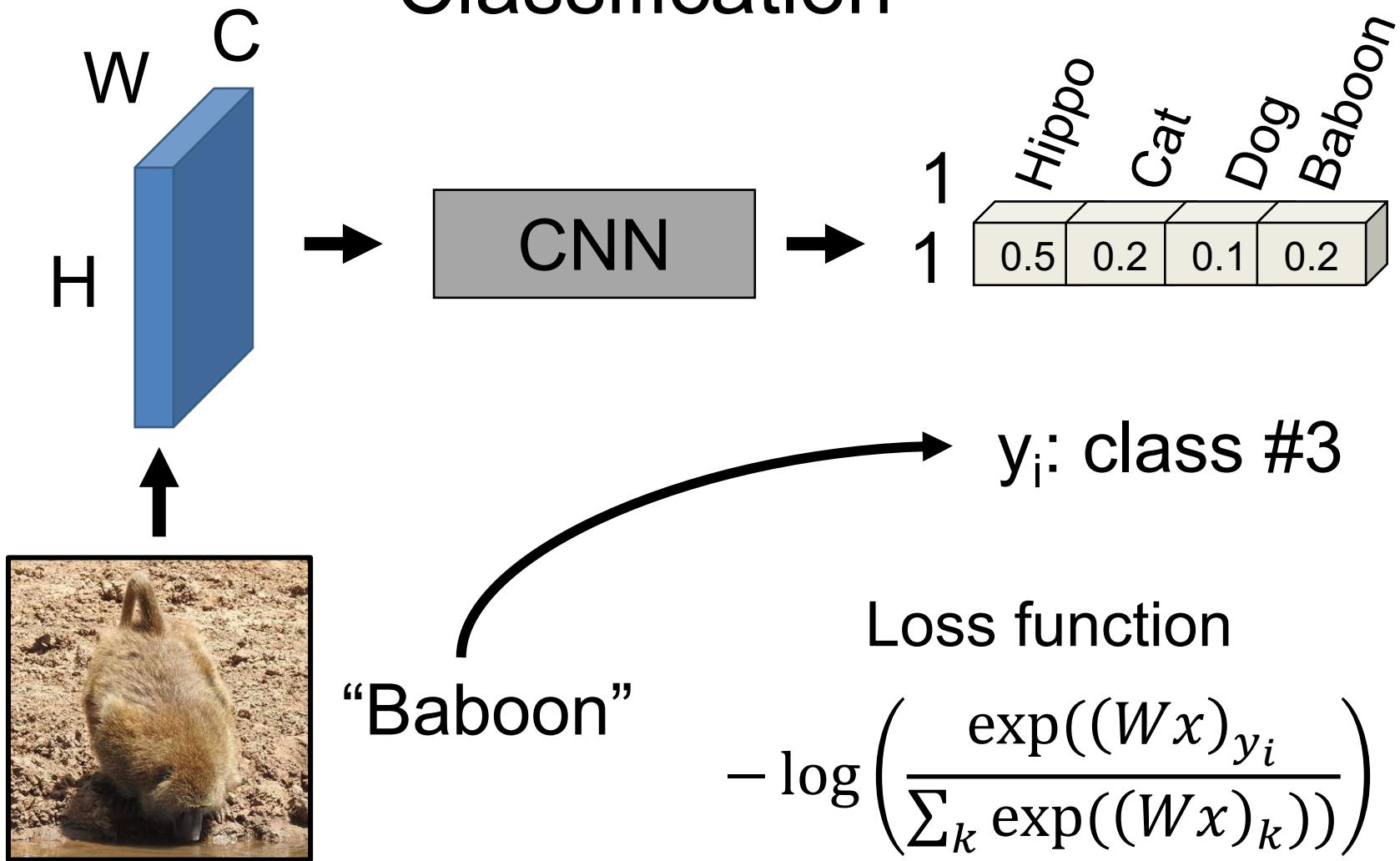
$P(\text{image is class } \#1)$
 $P(\text{image is class } \#2)$
 $P(\text{image is class } \#F)$



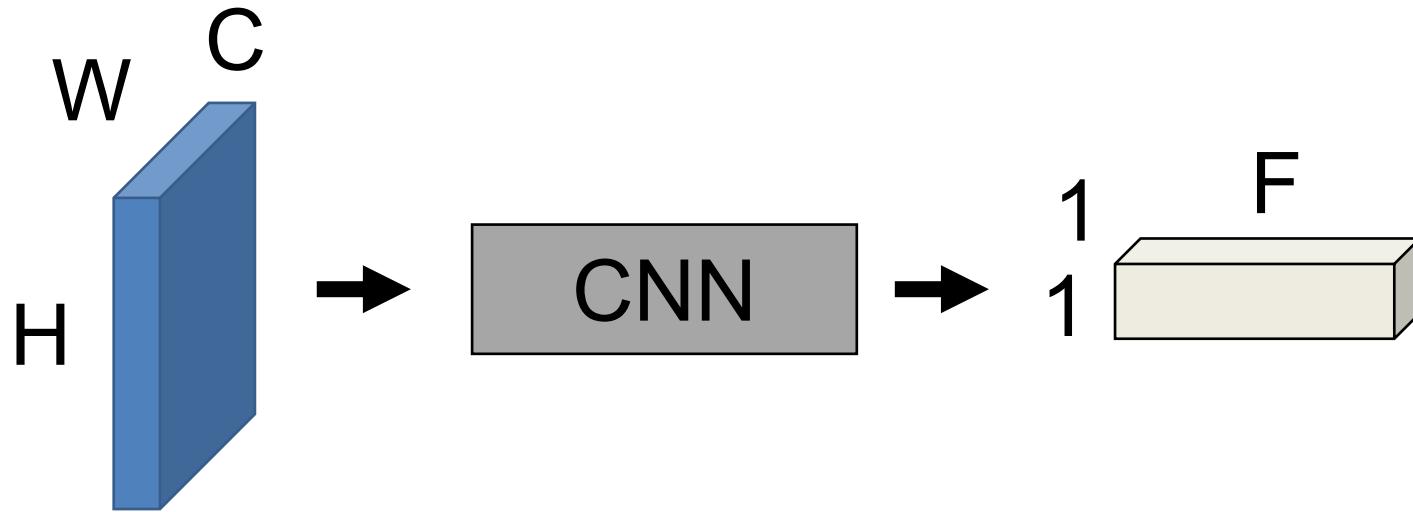
Today's Running Example: Classification



Today's Running Example: Classification



Model For Your Head



- Provide:
 - Examples of images and desired outputs
 - Sequence of layers producing a $1 \times 1 \times F$ output
 - A loss function that measures success
- Train the network -> network figures out the parameters that makes this work

Layer Collection

You can construct functions out of layers. The only requirement is the layers “fit” together. Optimization figures out what the parameters of the layers are.



Image credit: lego.com

Review – Pooling

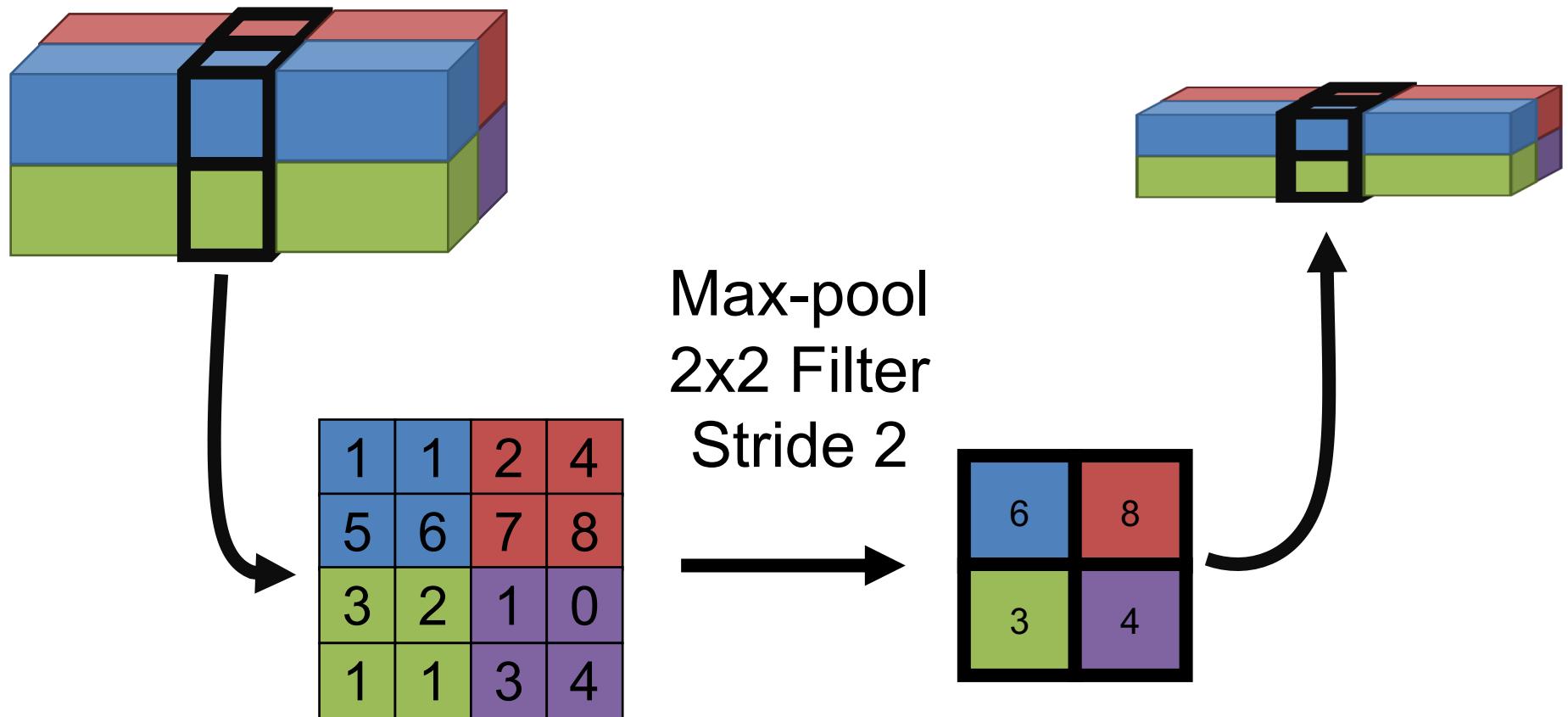
Idea: just want spatial resolution of activations
/ images smaller; applied per-channel

1	1	2	4
5	6	7	8
3	2	1	0
1	1	3	4

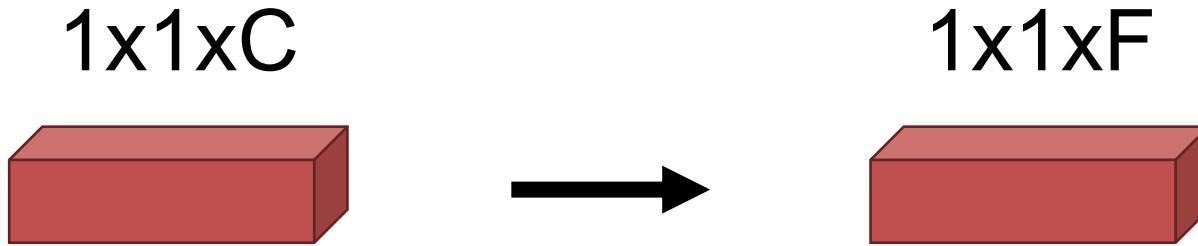
Max-pool
2x2 Filter
Stride 2


6	8
3	4

Review – Pooling



Other Layers – Fully Connected



Map C-dimensional feature to F-dimensional
feature using linear transformation
 W ($F \times C$ matrix) + b ($F \times 1$ vector)

How can we write this as a convolution?

Everything's a Convolution

1x1xC



1x1xF



Set Fh=1, Fw=1

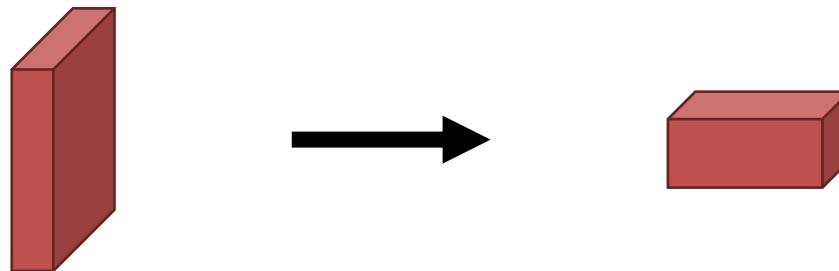
1x1 Convolution with F Filters

$$b + \sum_{i=1}^{F_h} \sum_{j=1}^{F_w} \sum_{k=1}^c F_{i,j,k} I_{y+i,x+j,k} \longrightarrow b + \sum_{k=1}^c F_k I_c$$

Converting to a Vector

HxWxC

1x1xF

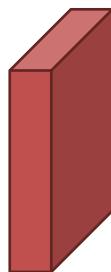


How can we do this?

Converting to a Vector* – Pool

$H \times W \times C$

$1 \times 1 \times F$



1	1	2	4
5	6	7	8
3	2	1	0
1	1	3	4

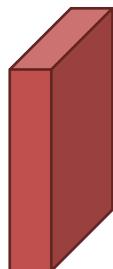
Avg Pool
 $H \times W$ Filter
Stride 1



$^*(\text{If } F == C)$

Converting to a Vector* – Pool

$H \times W \times C$



$1 \times 1 \times F$



1	1	2	4
5	6	7	8
3	2	1	0
1	1	3	4

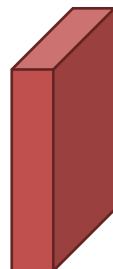
Avg Pool
 $H \times W$ Filter
Stride 1



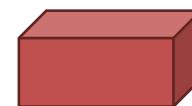
*(If $F \neq C$ apply a fully-connected layer afterward)

Converting to a Vector – Convolve

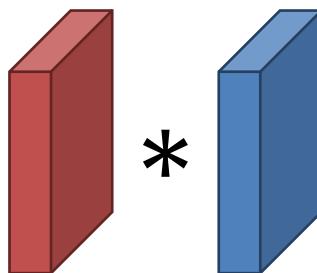
$H \times W \times C$



$1 \times 1 \times F$



$H \times W$ Convolution with F Filters



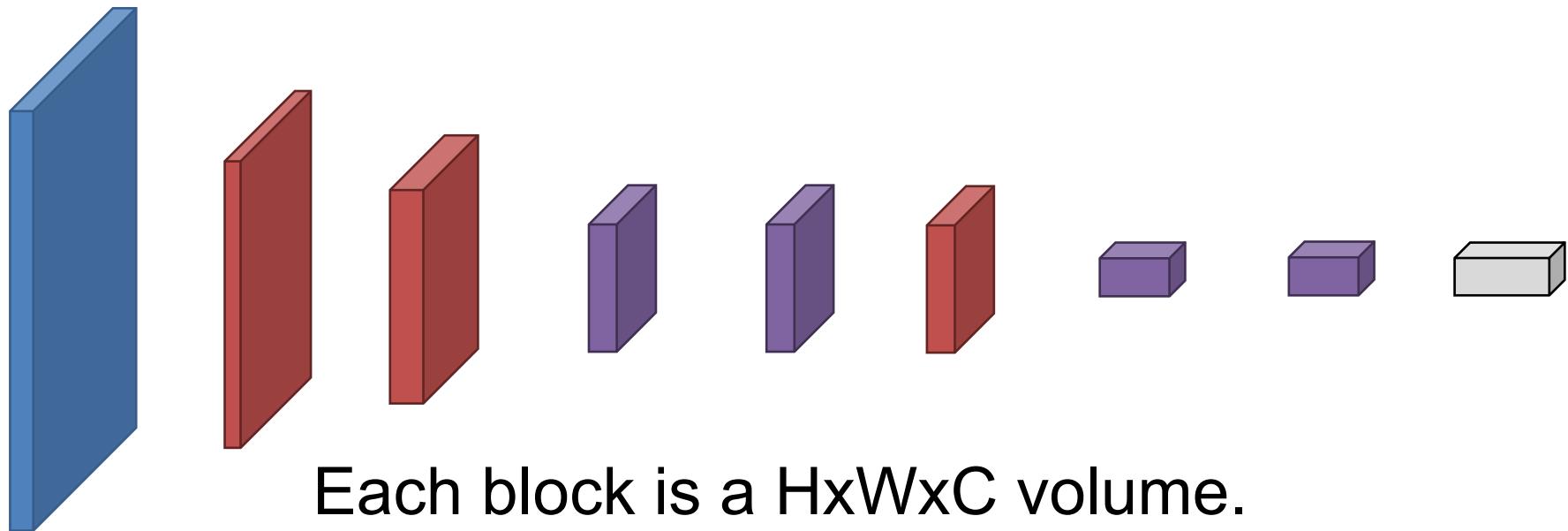
Single value
Per-filter

Looking At Networks

- We'll look at 3 landmark networks, each trained to solve a 1000-way classification output (Imagenet)
 - Alexnet (2012)
 - VGG-16 (2014)
 - Resnet (2015)

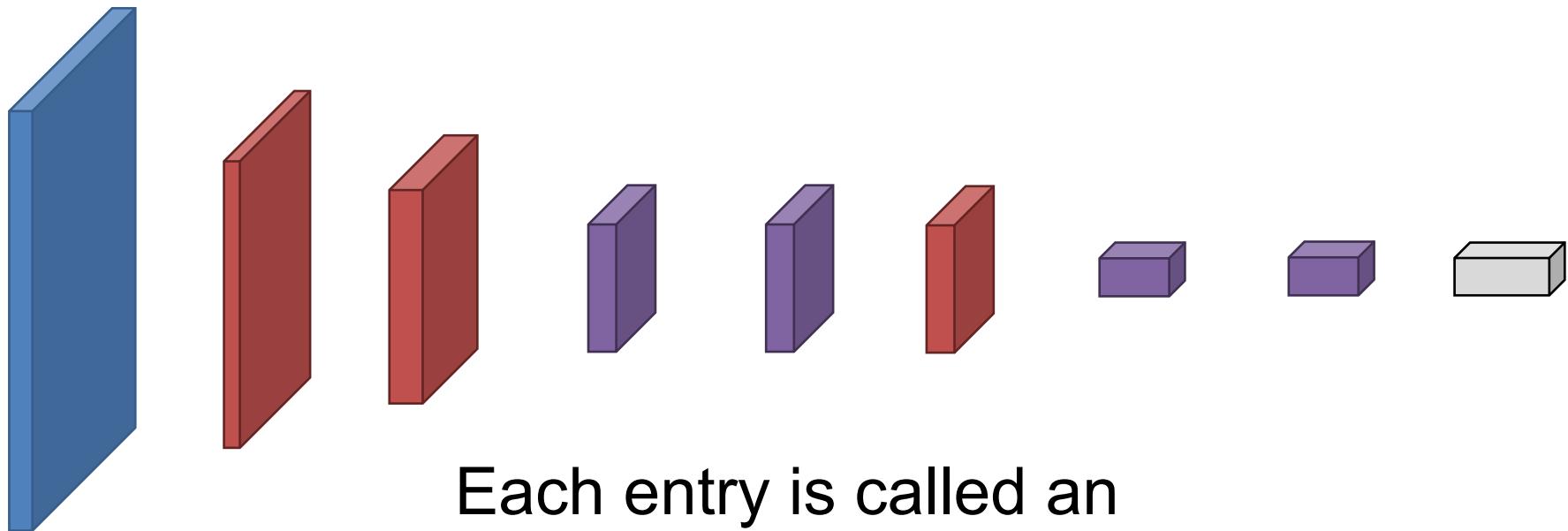
AlexNet

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



CNN Terminology

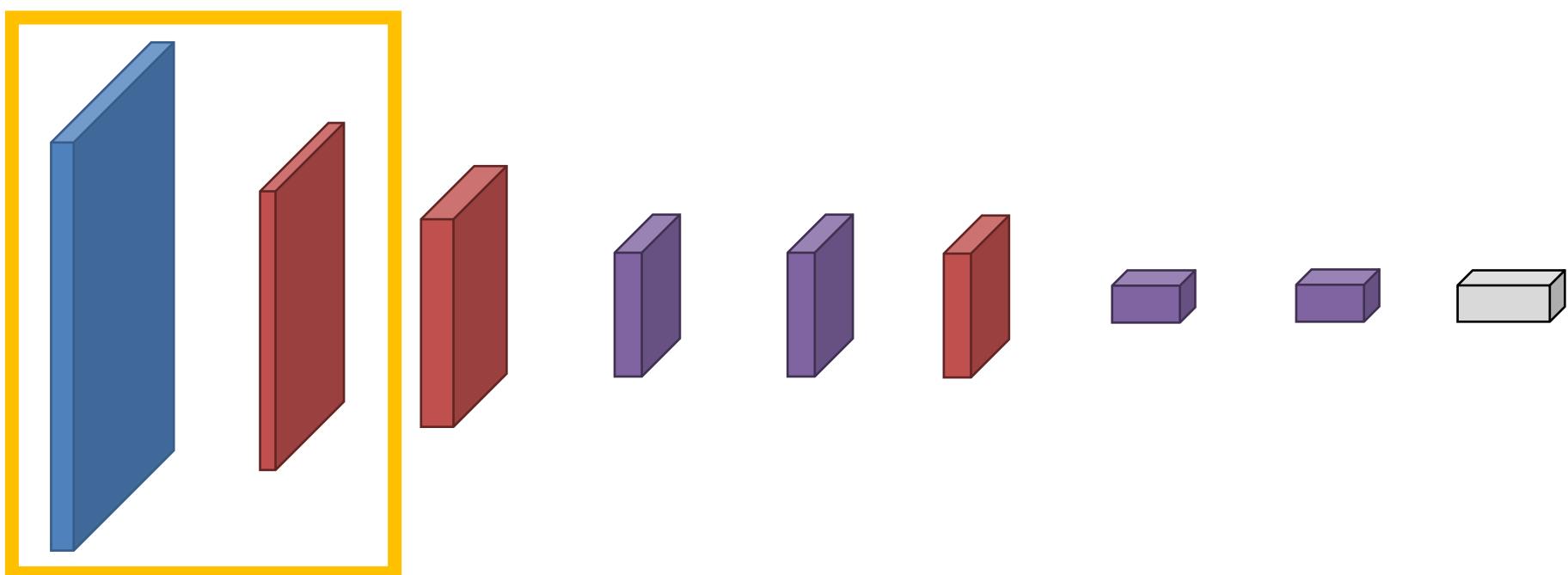
Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



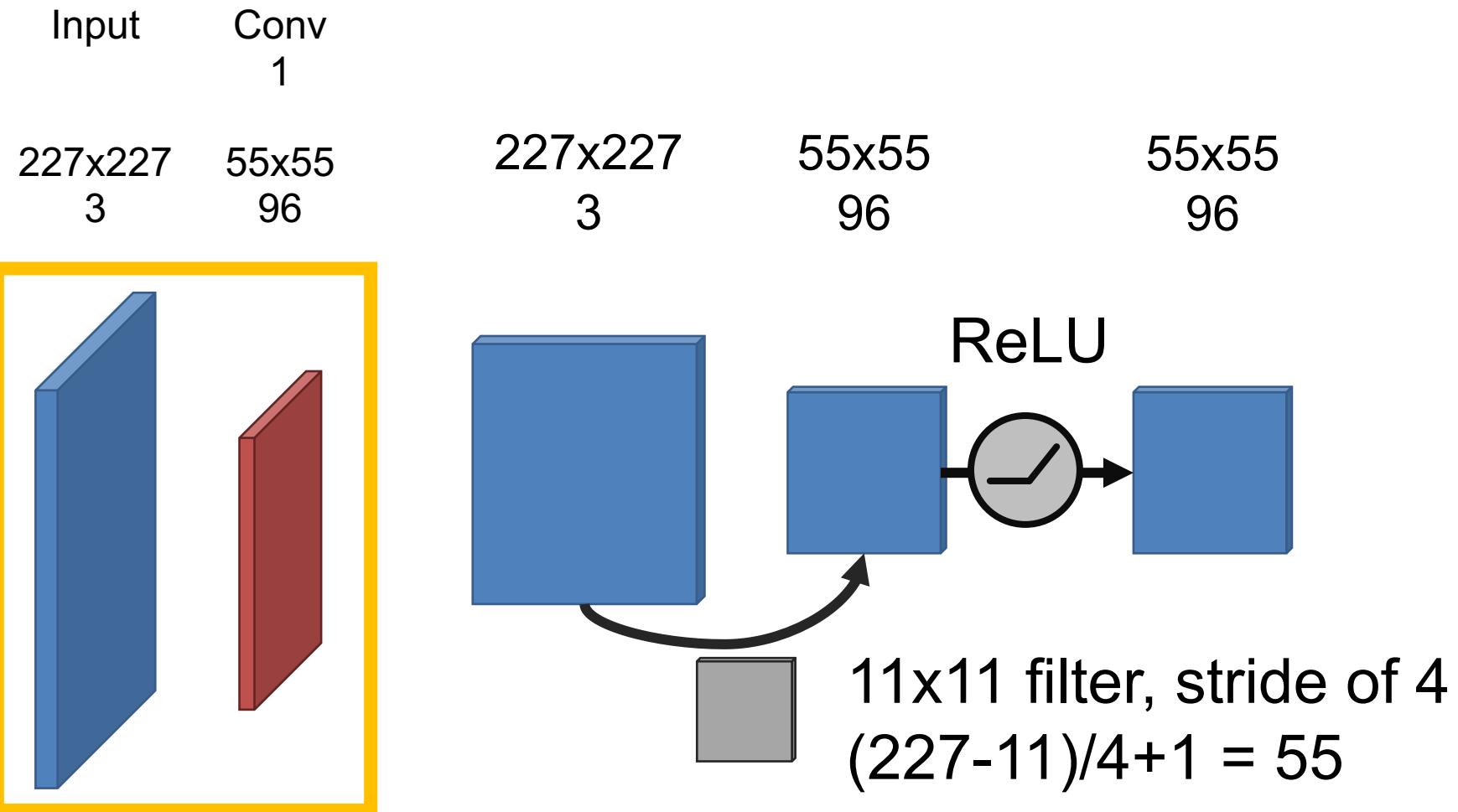
Each entry is called an
“activation”/“(intermediate) feature”/”

AlexNet

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000

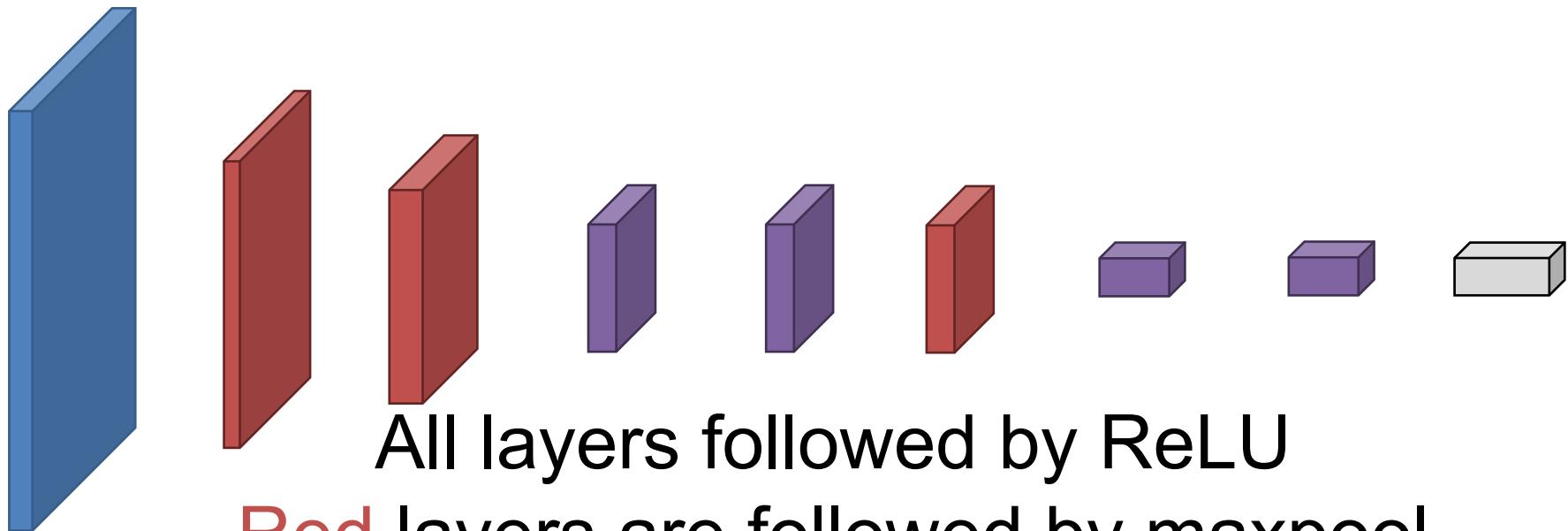


AlexNet



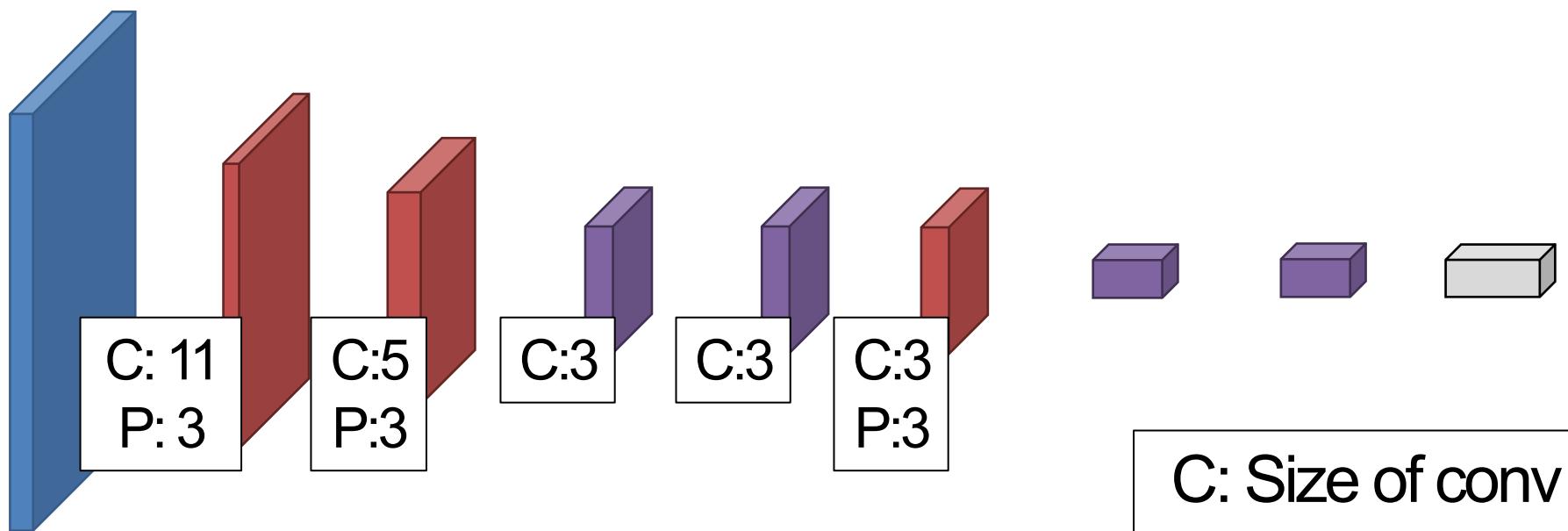
AlexNet

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



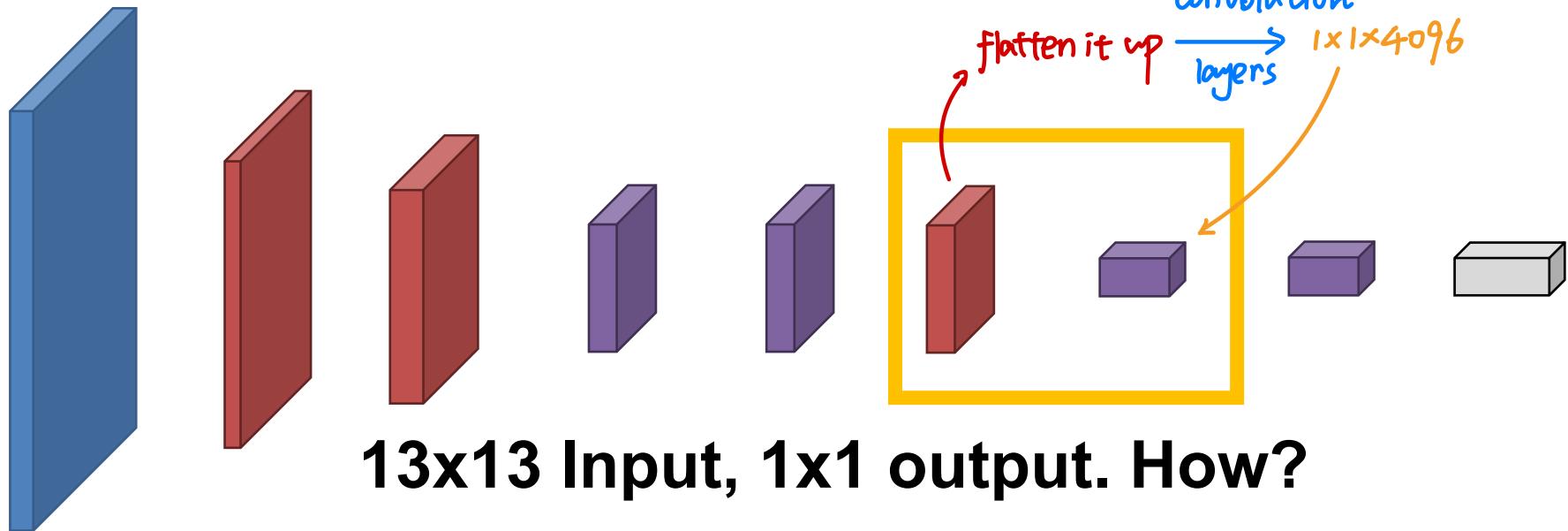
AlexNet – Details

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



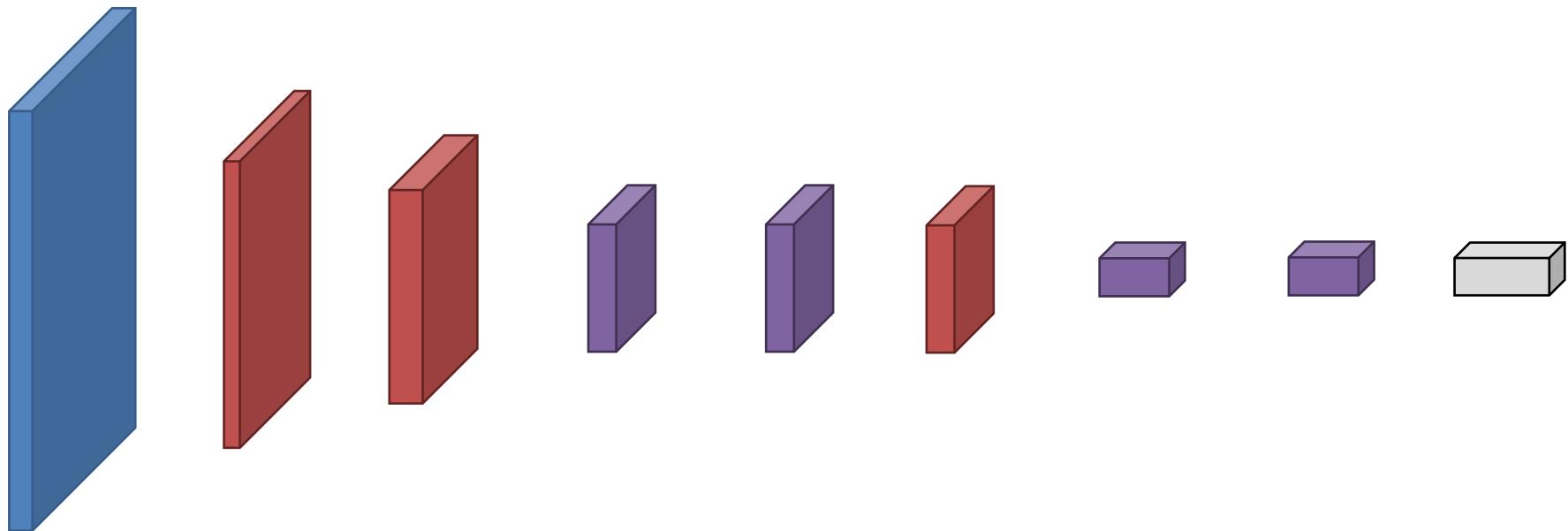
AlexNet

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



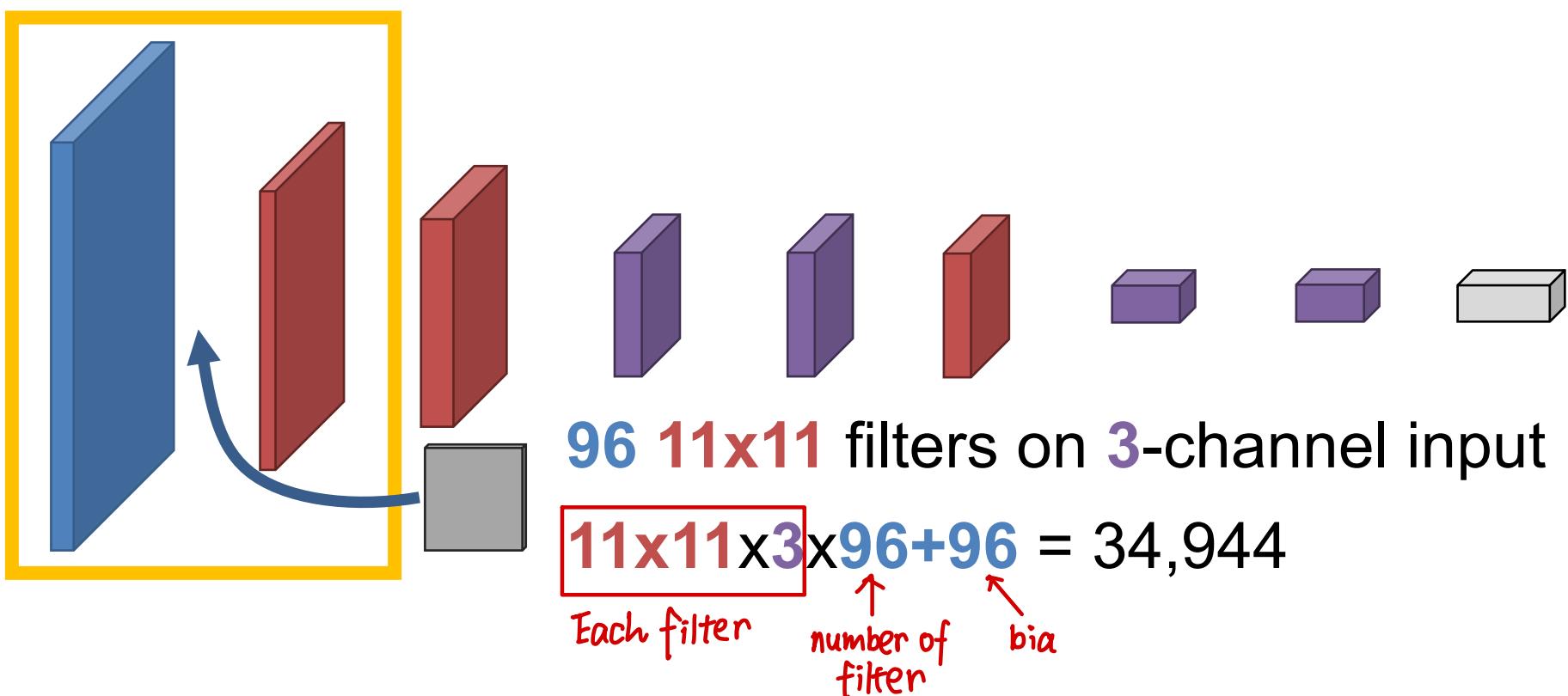
Alexnet – How Many Parameters?

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



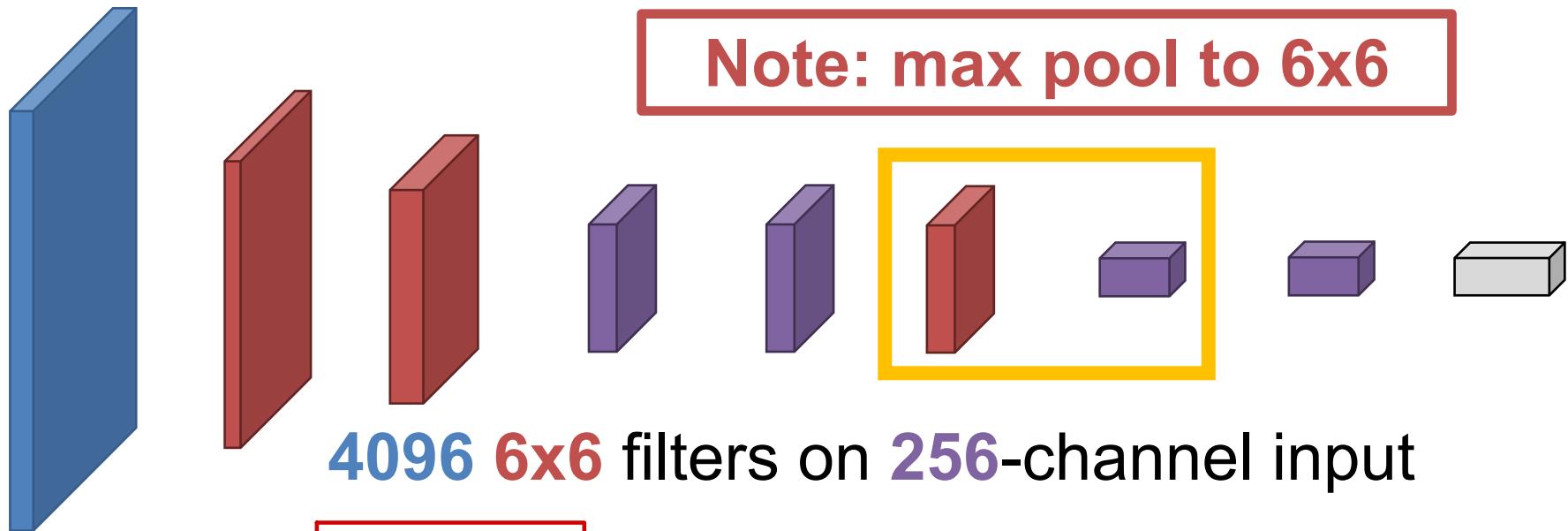
Alexnet – How Many Parameters?

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



Alexnet – How Many Parameters?

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



Note: max pool to 6x6

4096 6x6 filters on 256-channel input

$$6 \times 6 \times 256 \times 4096 + 4096 = 38 \text{ million}$$

each filter ↑ number of filter bias

Alexnet – How Many Parameters?

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



Alexnet – How Many Parameters

How long would it take you to list the parameters of Alexnet at 4s / parameter?

1 year?

4 years?

8 years?

16 years?

- 62.4 million parameters
- *Vast majority in fully connected layers*
 - But... paper notes that removing the convolutions is disastrous for performance.

Dataset – ILSVRC

- Imagenet Largescale Visual Recognition Challenge
- 1.4M images
- 1000 Categories, often ridiculously precise

Dataset – ILSVRC

birds



flamingo

cock

ruffed grouse

quail

partridge

...

bottles



pill bottle

beer bottle

wine bottle

water bottle

pop bottle

...

cars



race car

wagon

minivan

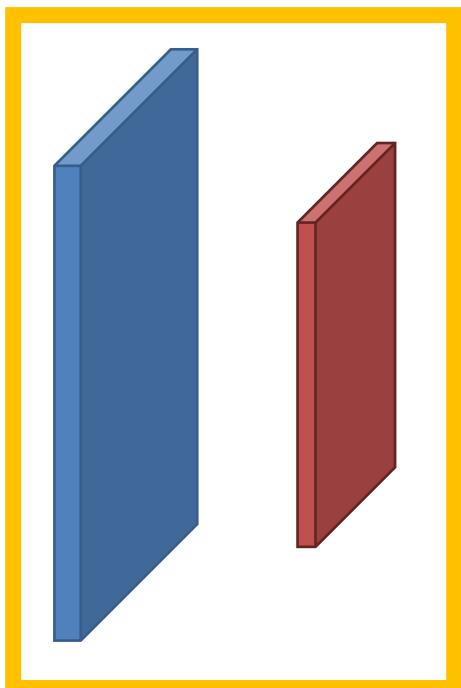
jeep

cab

...

Visualizing Filters

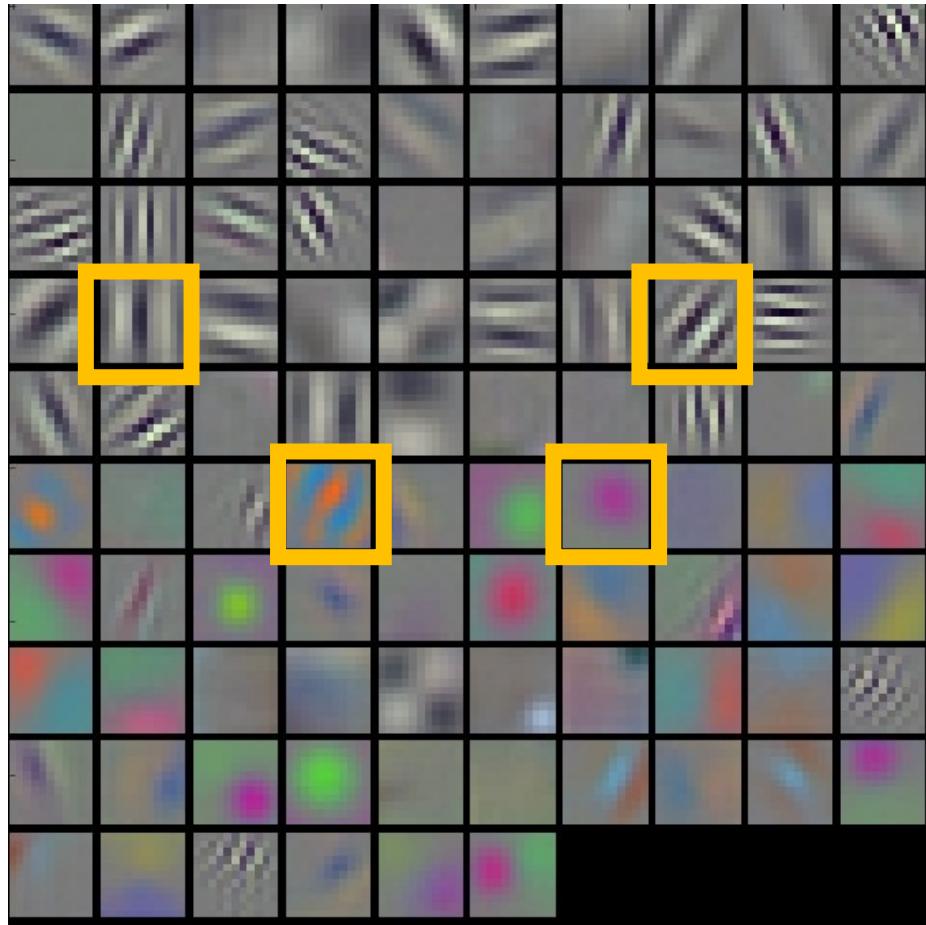
Input	Conv
227x227	1
3	55x55
	96



Conv 1 Filters

- **Q. How many input dimensions?**
 - A: 3
- **What does the input mean?**
 - R, G, B, duh.

What's Learned – Recap

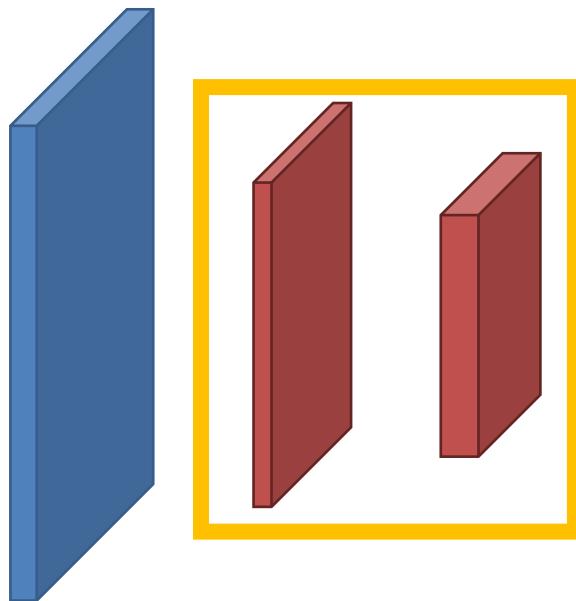


First layer filters of a network trained to distinguish 1000 categories of objects

Remember these filters go over color.

Visualizing Later Filters

Input	Conv 1	Conv 2
227x227	55x55	27x27
3	96	256



Conv 2 Filters

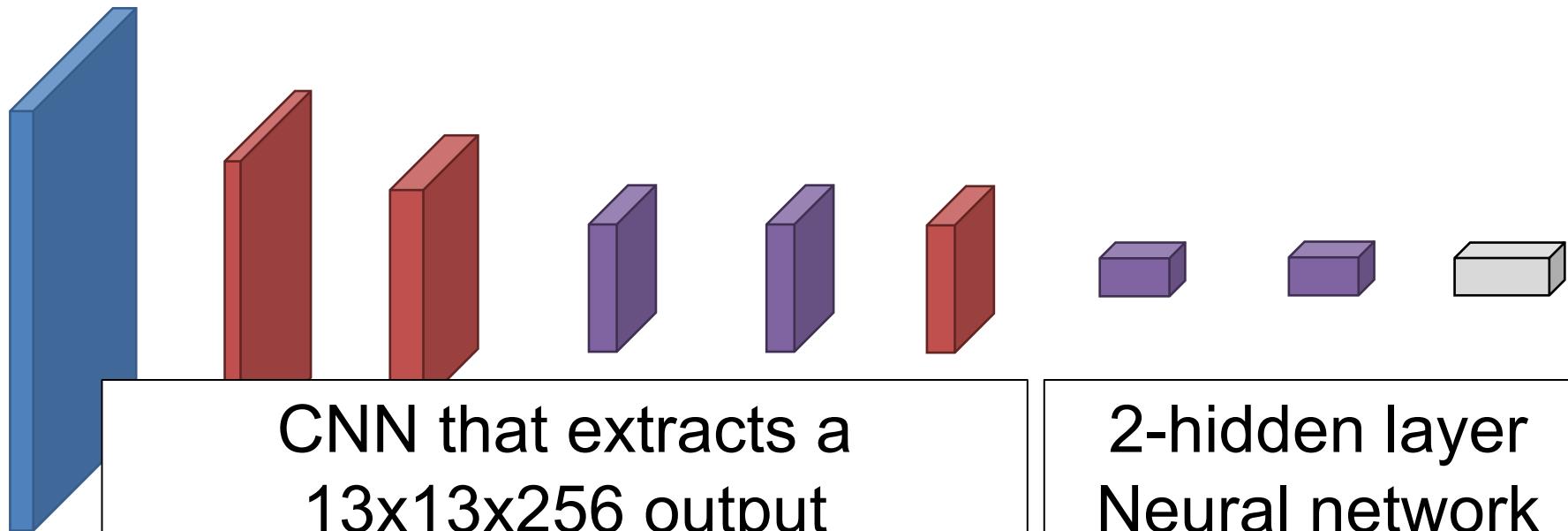
- **Q. How many input dimensions?**
 - A: 96.... hmmm
- **What does the input mean?**
 - Uh, the uh, previous slide

Visualizing Later Filters

- Understanding the meaning of the later filters *from their values* is typically impossible: too many input dimensions, not even clear what the input means.

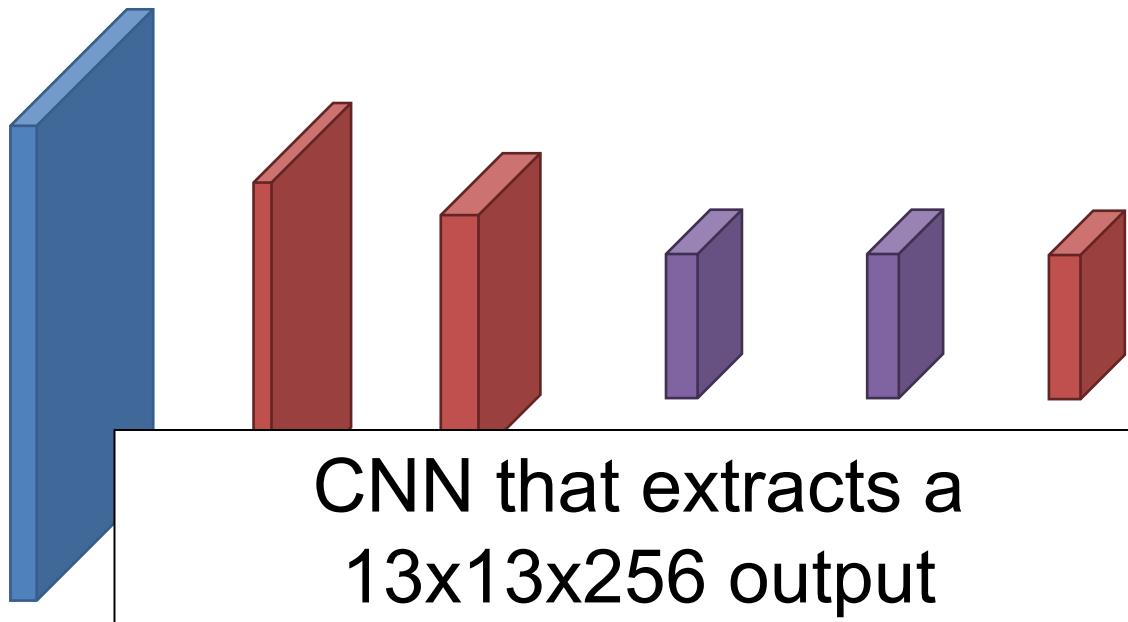
Understanding Later Filters

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



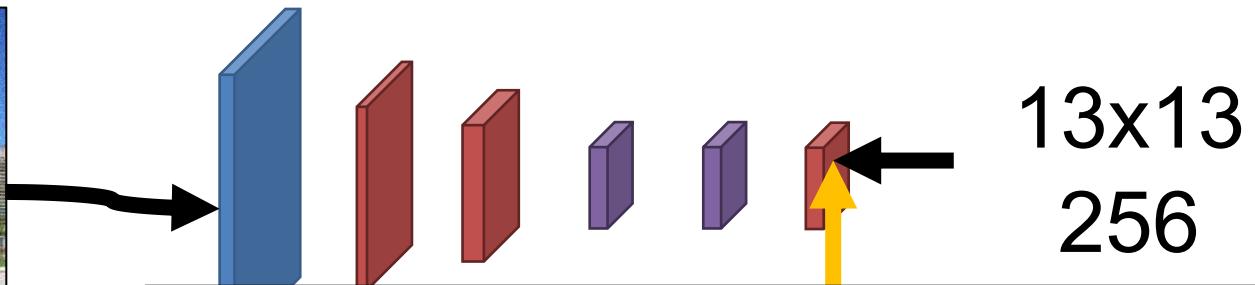
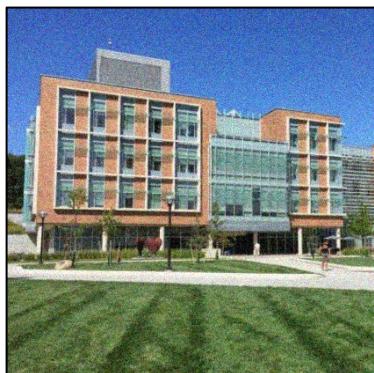
Understanding Later Filters

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256

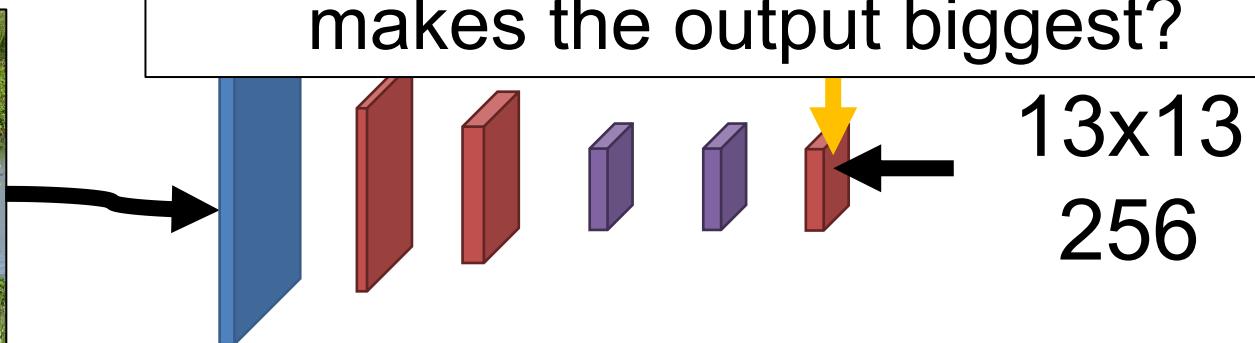


Understanding Later Filters

Feed an image in, see what score the filter gives it. A more pleasant version of a real neuroscience procedure.



Which one's bigger? What image makes the output biggest?



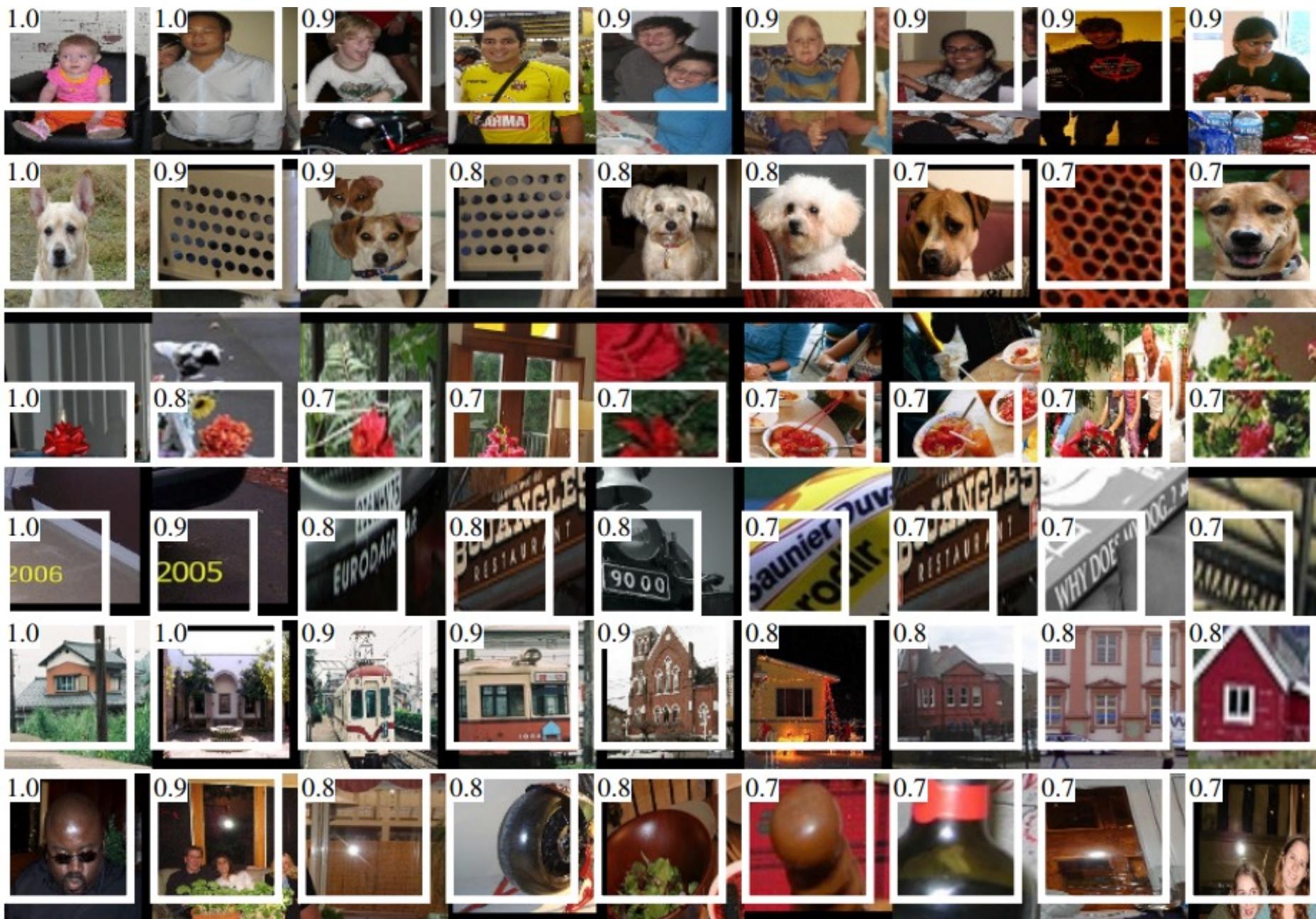
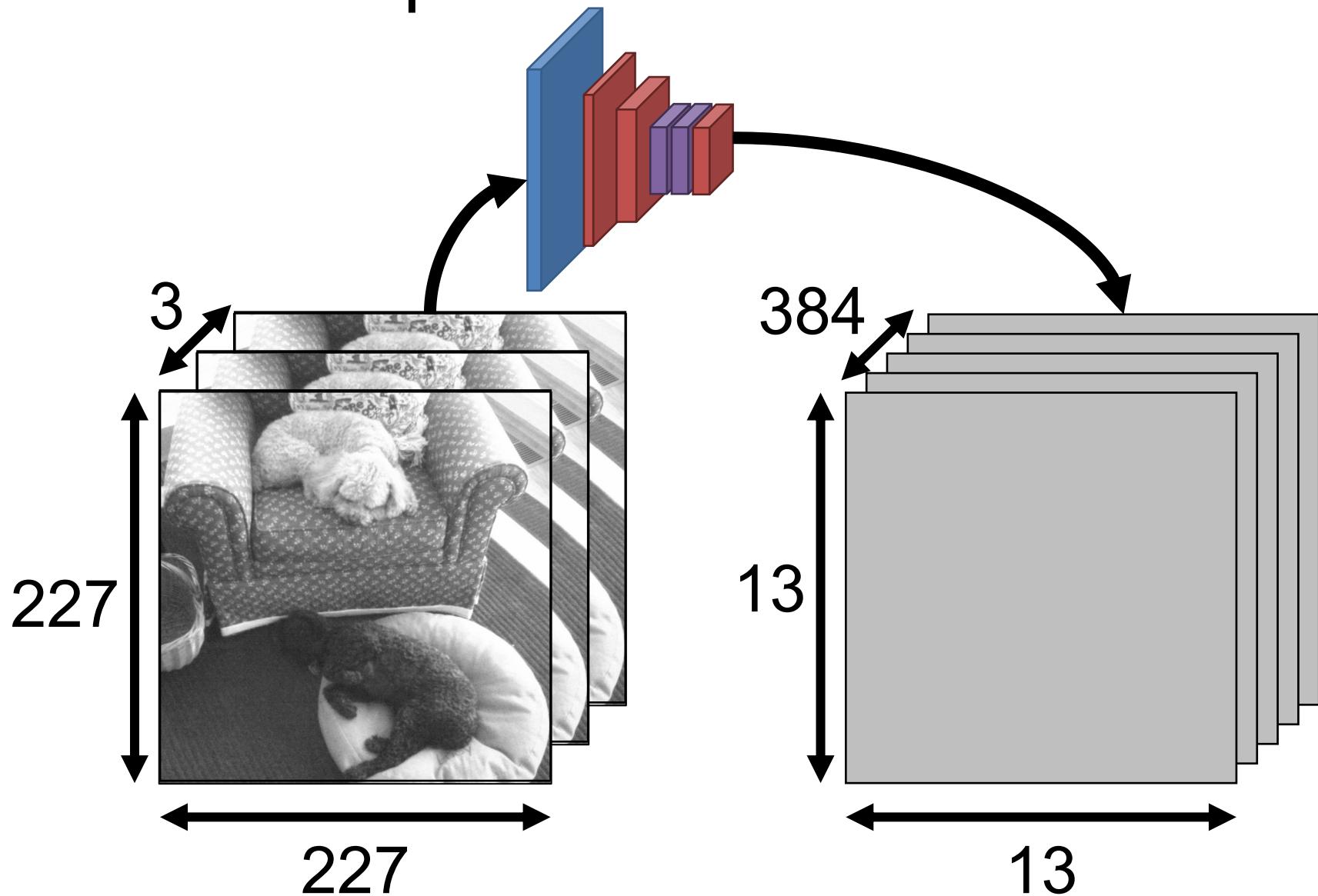
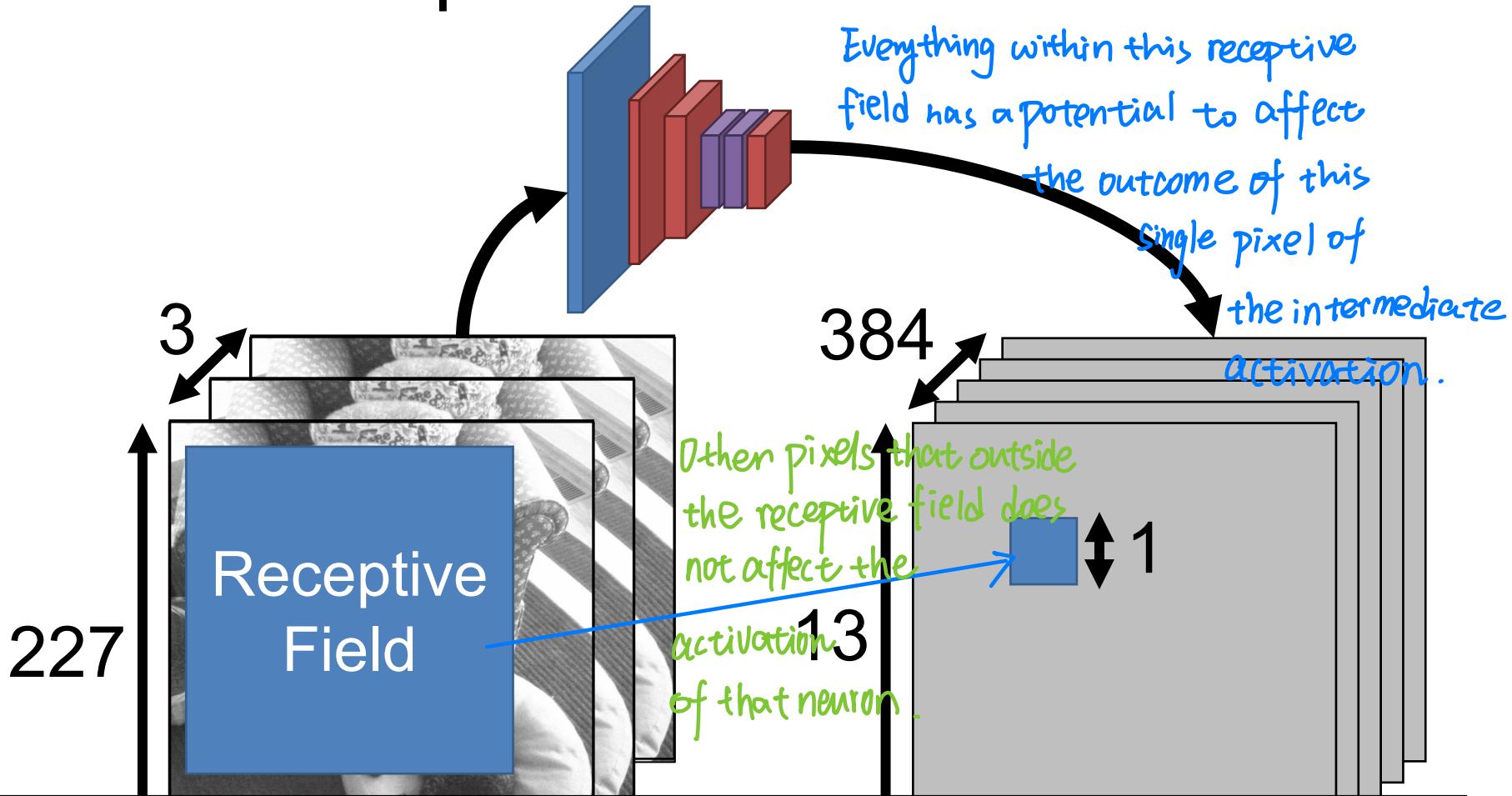


Figure Credit: Girschick et al. CVPR 2014.

What's Up With the White Boxes?



What's Up With the White Boxes?



Due to convolution, each later layer's value depends on / “sees” only a fraction of the input image.

Can use receptive fields to see where the network is “looking” to make its decisions

prison

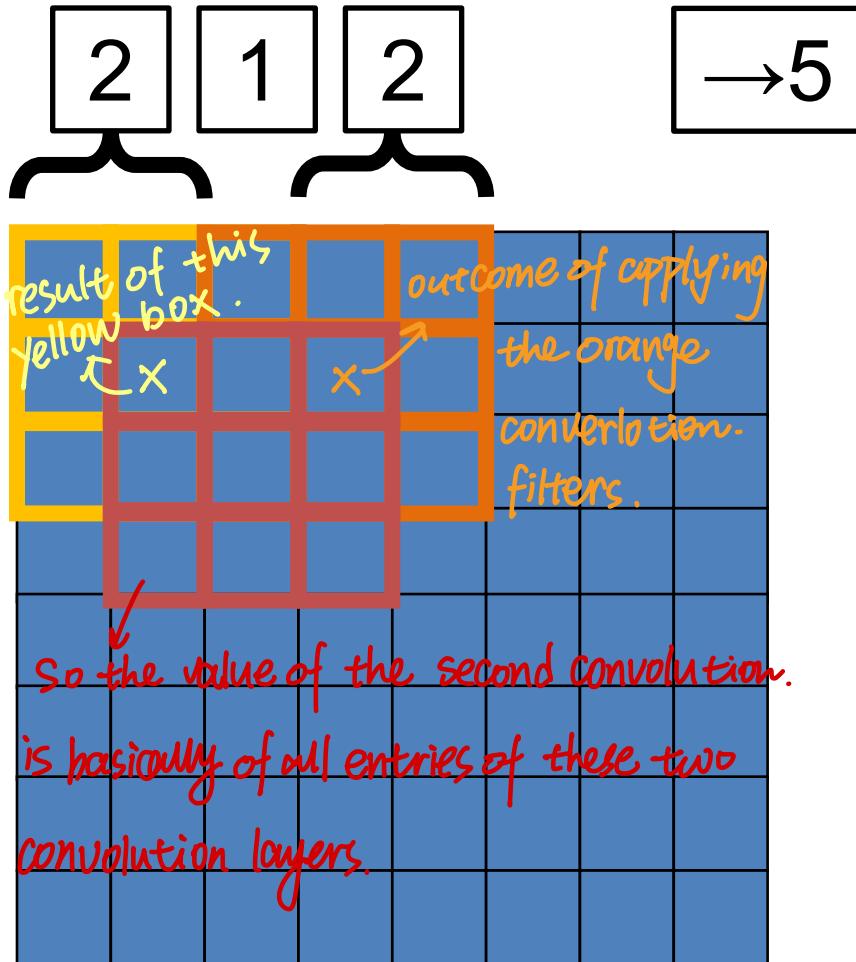


A very active area of research
(lots of great work done by Bolei Zhou, MIT now UCLA)

3 Tricks

- 3x3 Filters
- Batch Normalization
- Residual Learning

Key Idea – 3x3 Filters



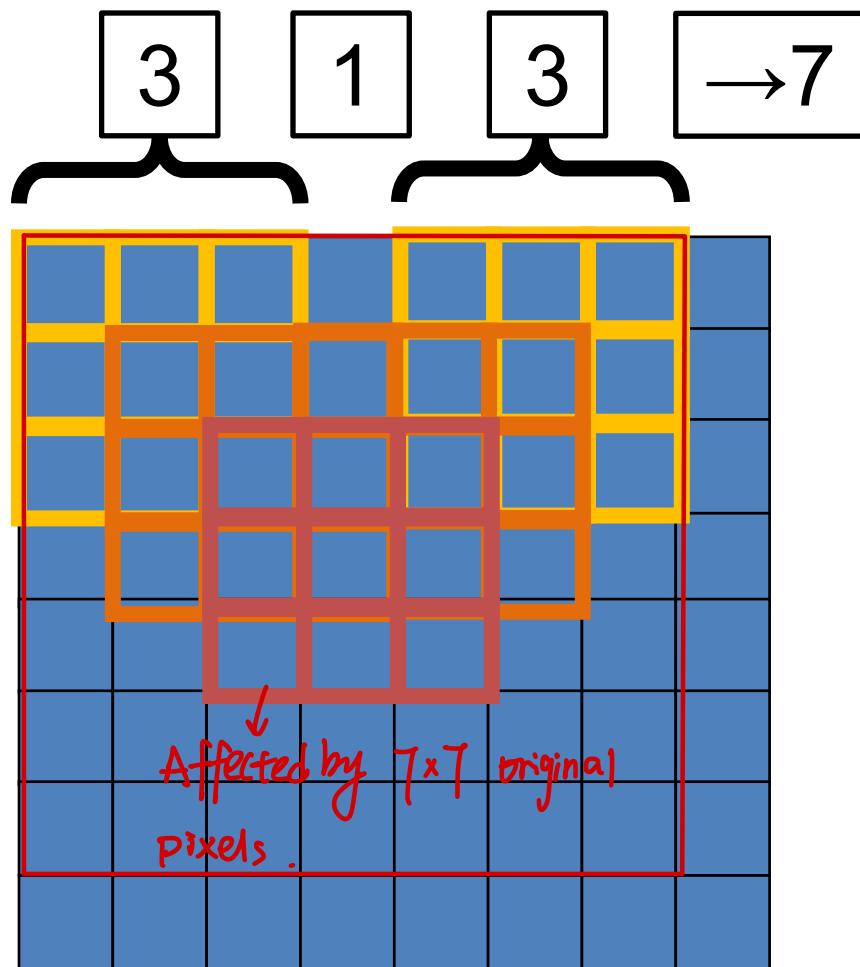
So if you apply this 3×3 filters twice.
then you combining the information
of these five entries. 5×5 area

3×3 filter followed by
 3×3 filter



Filter with 5×5
receptive field

Key Idea – 3x3 Filters



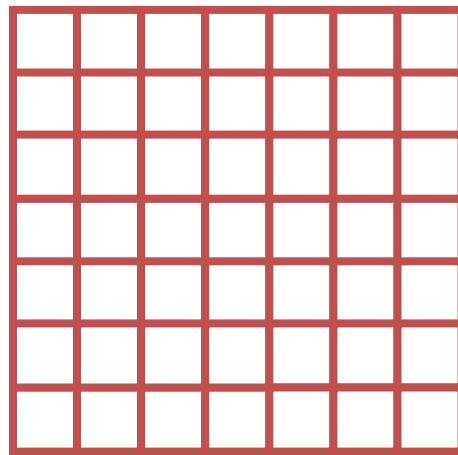
three series of sliding window convolution

3x3 filter followed by
3x3 filter followed by
3x3 filter



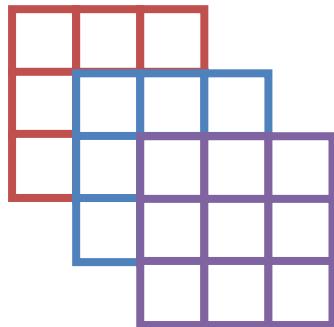
Filter with 7x7
receptive field

Why Does This Make A Difference?



Empirically, repeated 3×3 filters do better compared to a 7×7 filter.

Why?

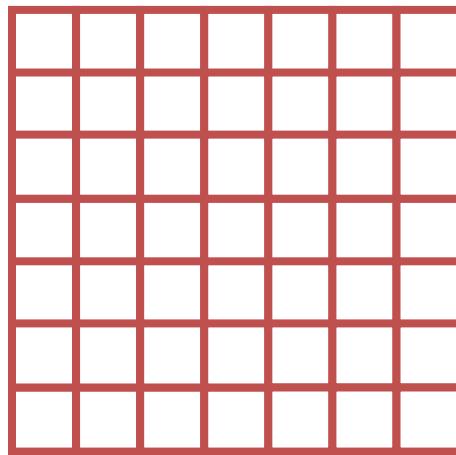


Better.

Reason: ① more parameter efficient.

Reason: ② we want more time of non-linearity

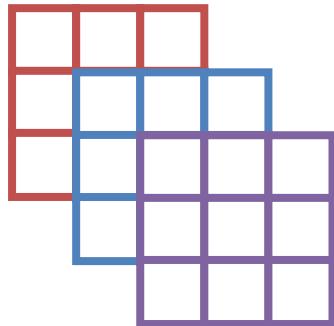
Key Idea – 3x3 Filters



Receptive Field: 7x7 pixels

Parameters/channel: 49

Number of ReLUs: 1



Receptive Field: 7x7 pixels

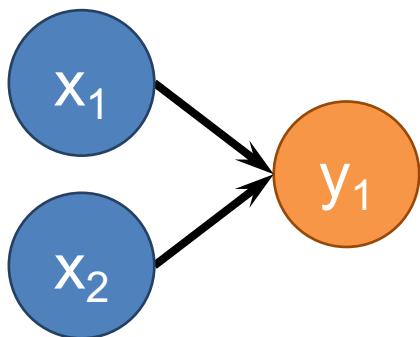
Parameters/channel: $3 \times 3 \times 3 = 27$

Number of ReLUs: 3

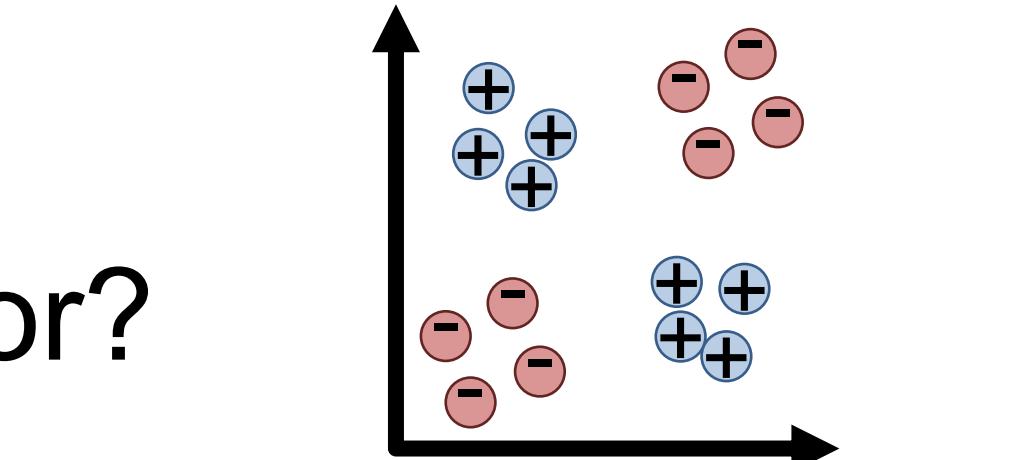
more non-linearity \longrightarrow more complex logic.

We Want More Non-linearity!

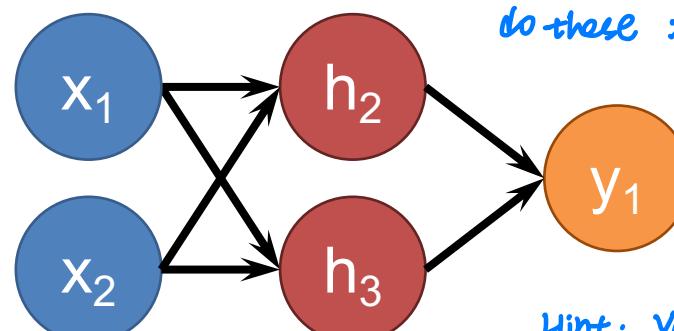
Can they
implement xor?



No



You can use the next layer to
do these xor operation.

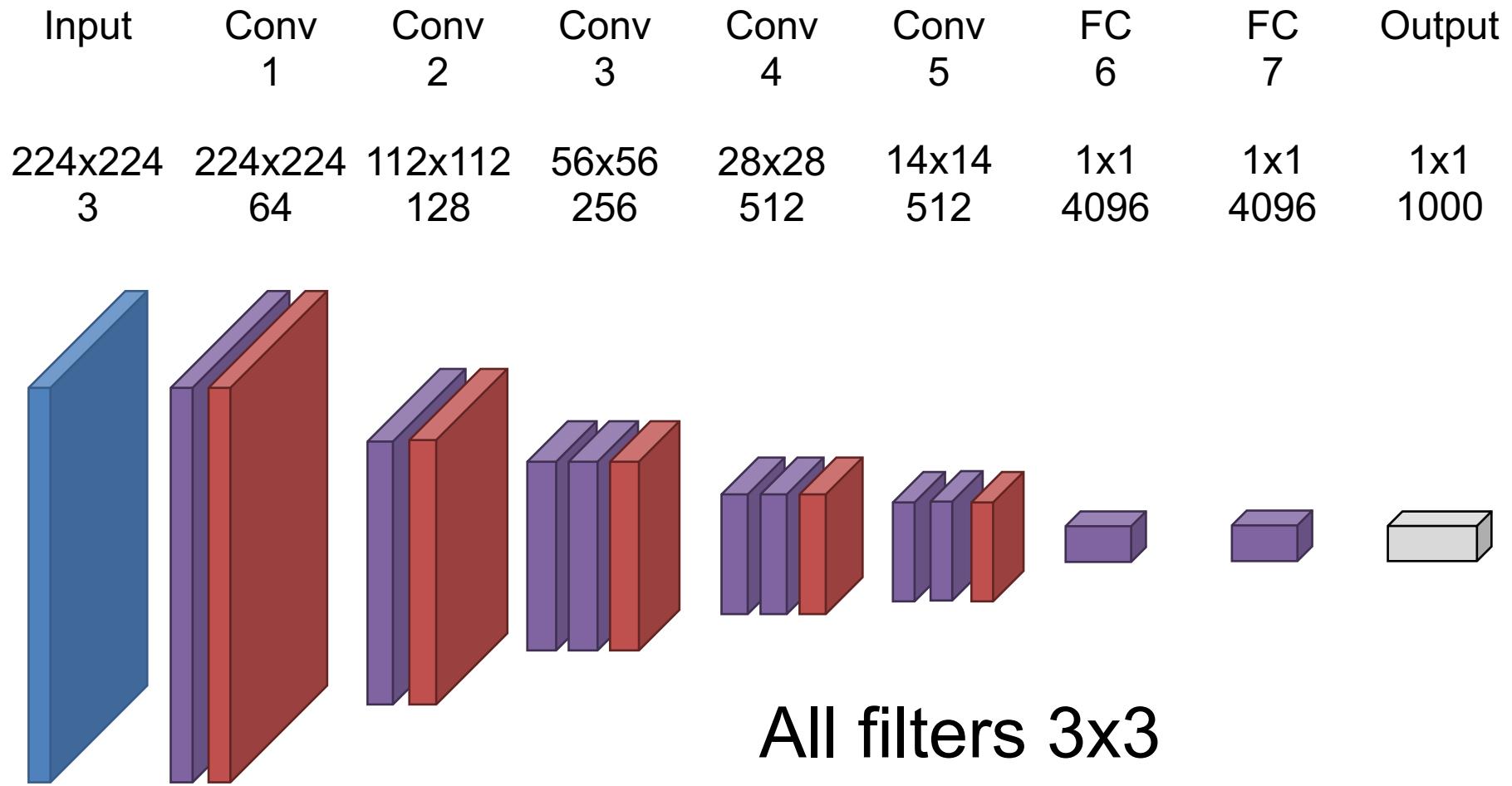


Yes

Possible with one hidden layer!
Hint: given two points, classify +,- negative).

Hint: You can use the first
hidden layer to do a classification
(whether this point is plus or
negative).

VGG16

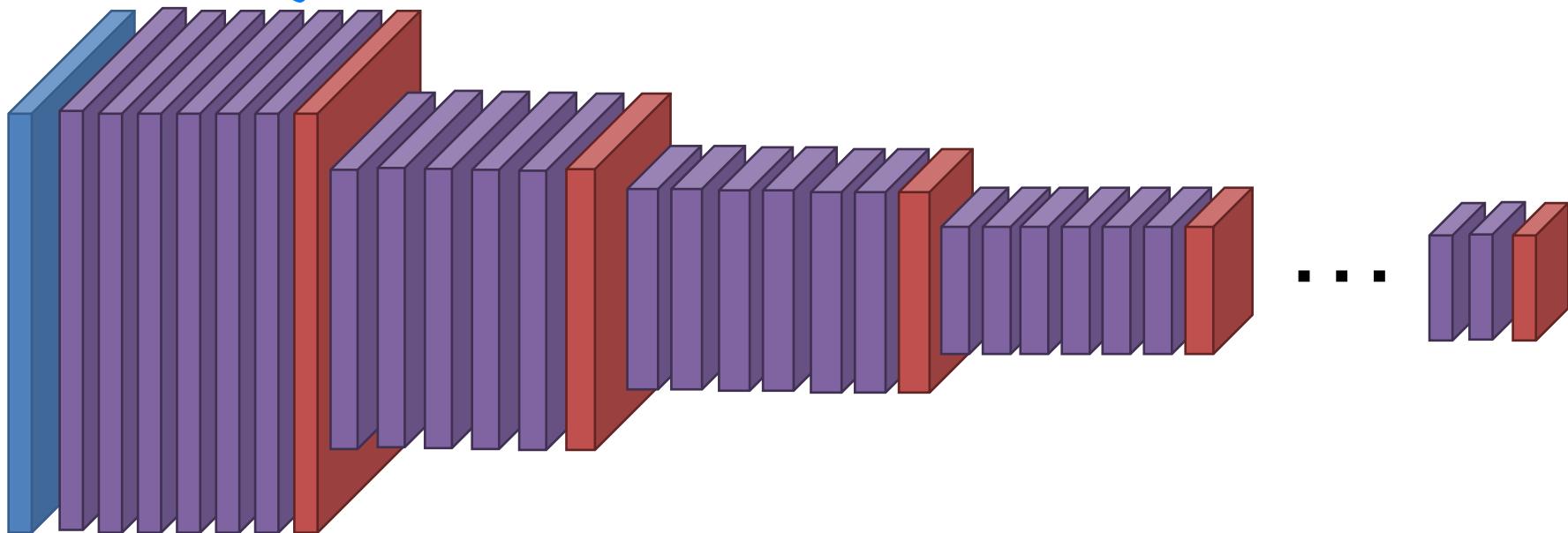


Training Deeper Networks

Why not just stack continuously?

What will happen to gradient going back?

The bottleneck that prevented us from doing this
is that there was something wrong with the
gradient going backward.



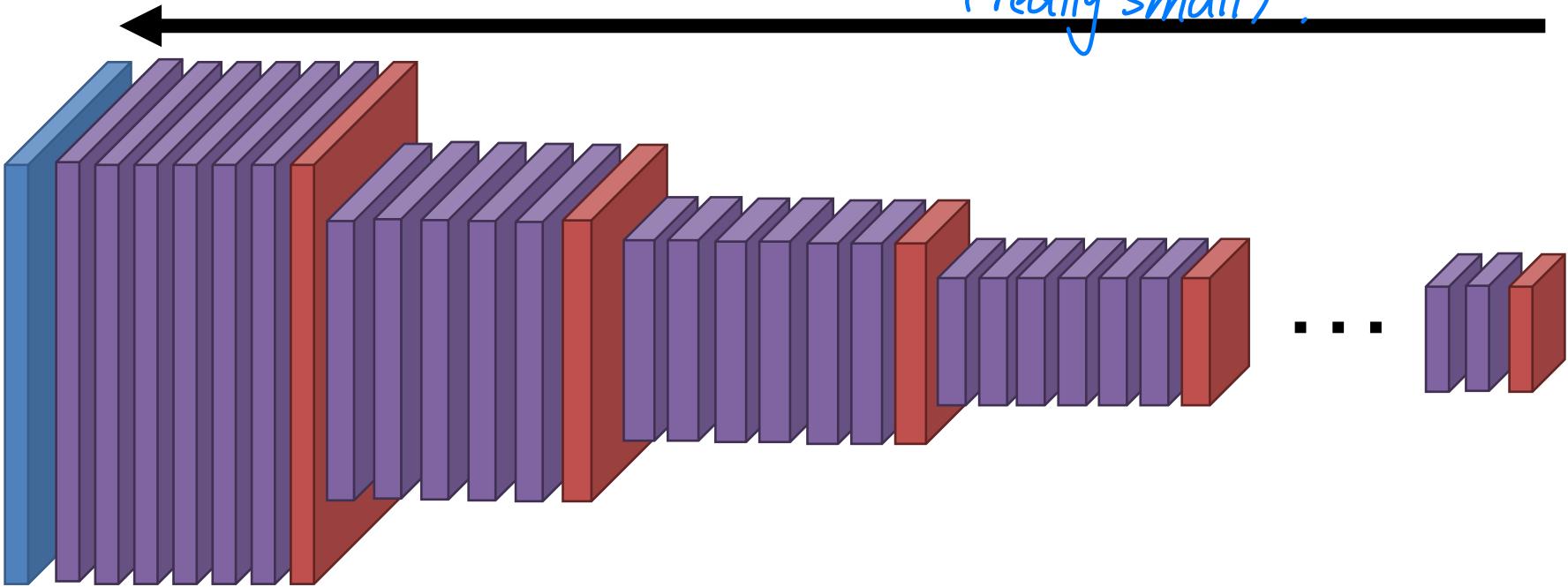
Backprop

Every backpropagation step multiplies
the gradient by the local gradient

$$1 * d * d * d \dots * d = d^{n-1}$$

What if $d < 1$, n big?

Vanishing Gradients
(really small).



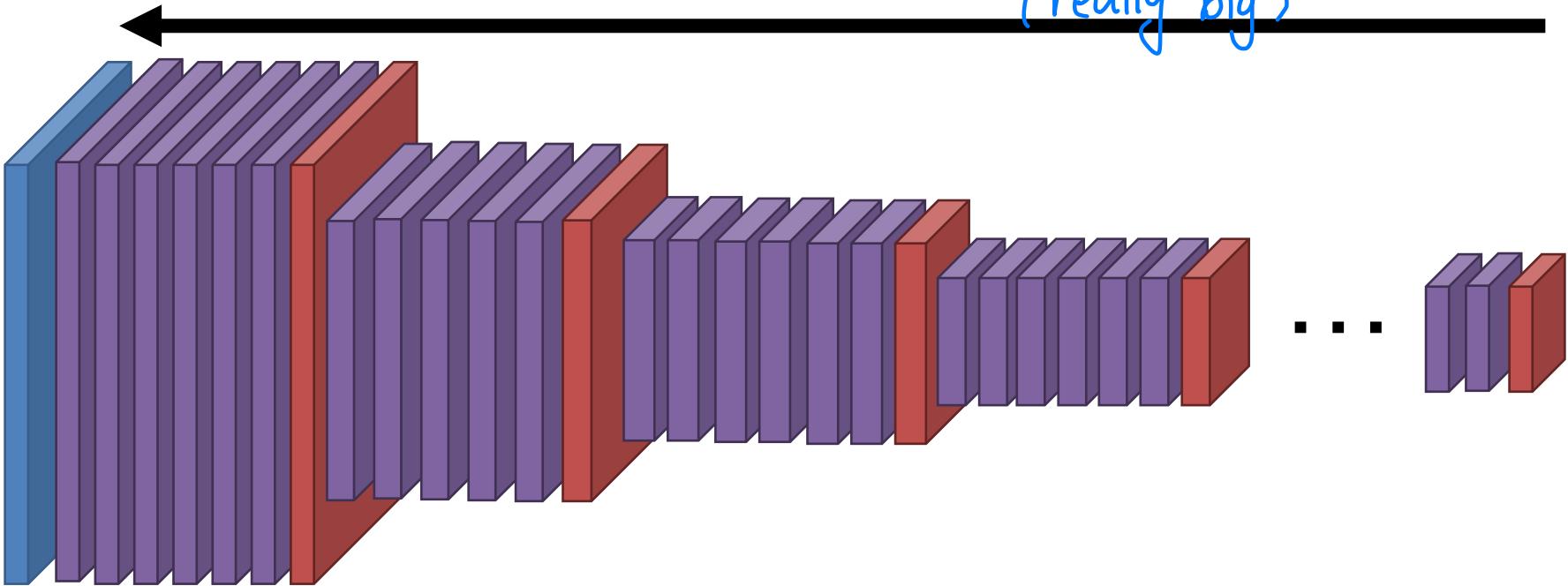
Backprop

Every backpropagation step multiplies
the gradient by the local gradient

$$1 * d * d * d \dots * d = d^{n-1}$$

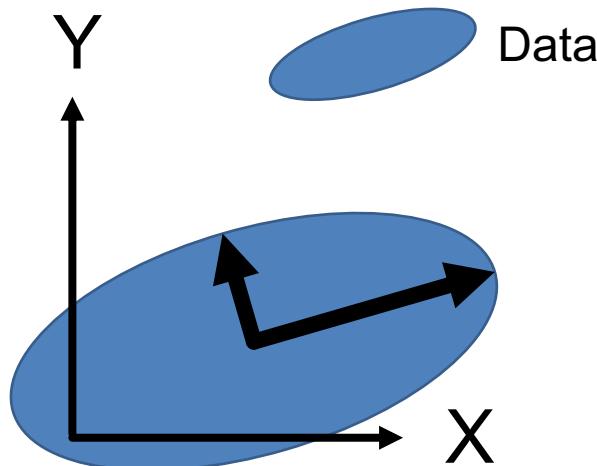
What if $d > 1$, n big?

Exploding Gradients
(really big)

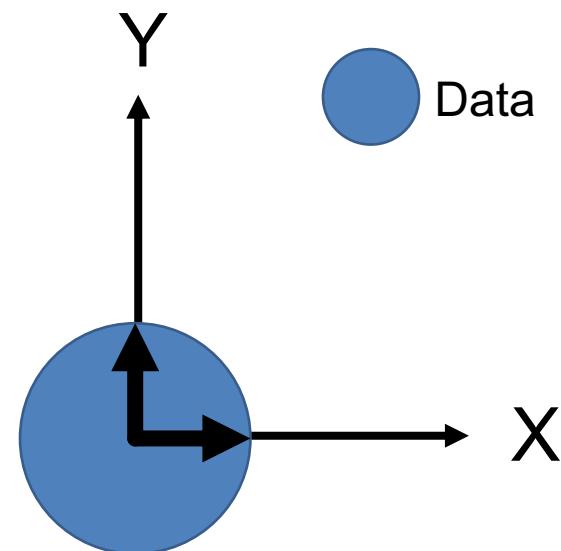


Solution 1 – Batch Normalization

Learning algorithms work far better when data looks like the right as opposed to the left



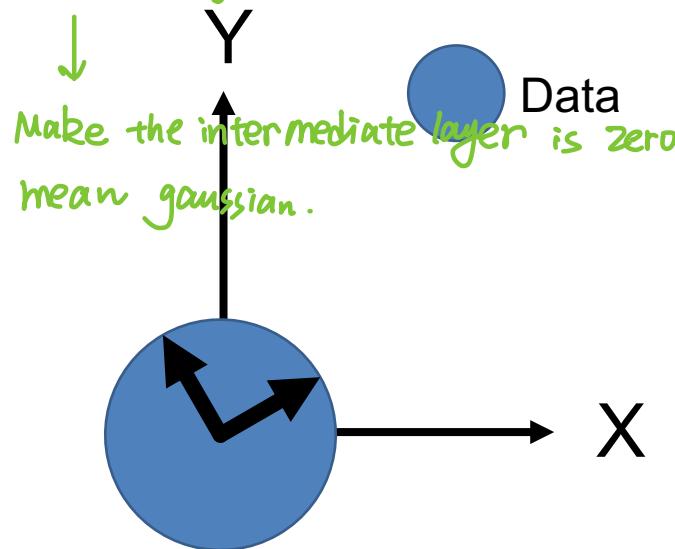
$$\begin{aligned}\text{Mean}(x) &\neq \text{Mean}(Y) \neq 0 \\ \text{Var}(x) &\neq \text{Var}(y) \neq 0 \\ \text{Cov}(x,y) &\neq 0\end{aligned}$$



$$\begin{aligned}\text{Mean}(x) &= \text{Mean}(Y) = 0 \\ \text{Var}(x) &= \text{Var}(y) = 1 \\ \text{Cov}(x,y) &= 0\end{aligned}$$

Solution 1 – Batch Normalization

normalize its input by estimating the variance of each batch.



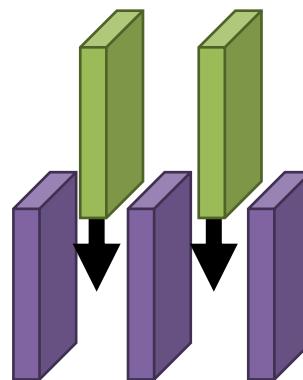
introduce new layer called: Batch Normalization layer.

Idea: make layer (**Batch Norm**) that normalizes things going through it based on estimates of $\text{Var}(x_i)$ in each batch.

Stick in between **other layers**
Source of tons of bugs

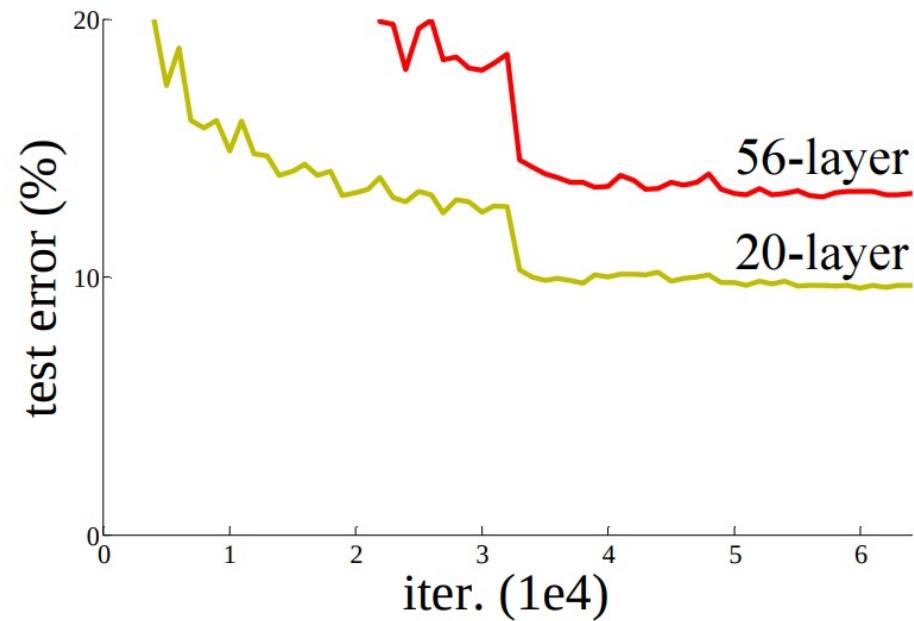
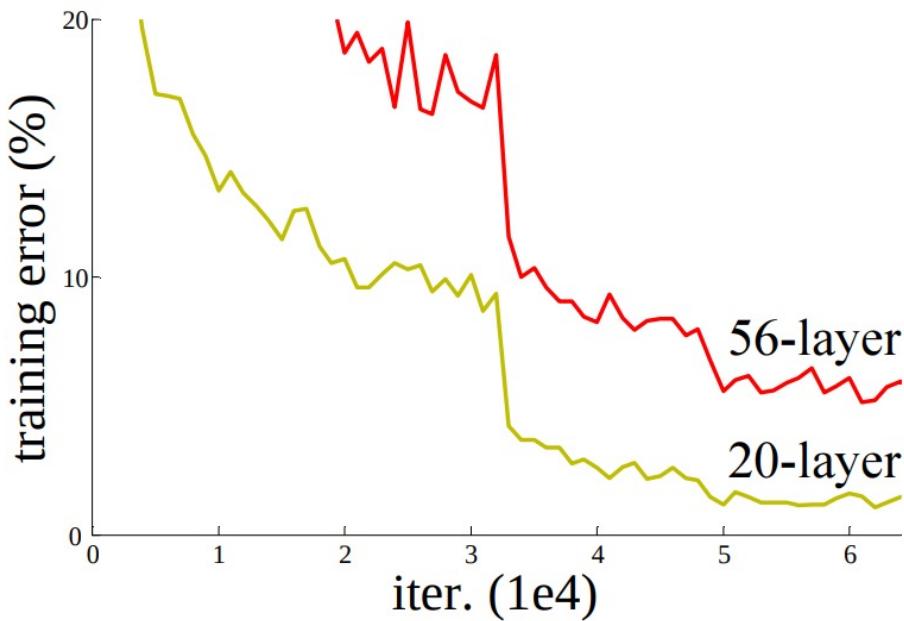
$$\text{Mean}(x) = \text{Mean}(Y) = 0$$

$$\text{Var}(x) = \text{Var}(y) = 1$$



There exists vs. We Can Find

- Still can't **fit** models to the data: **Deeper model** fits worse than **shallow model** on the training data.
- **There exists a deeper model that's identical to the shallow model. Why?**

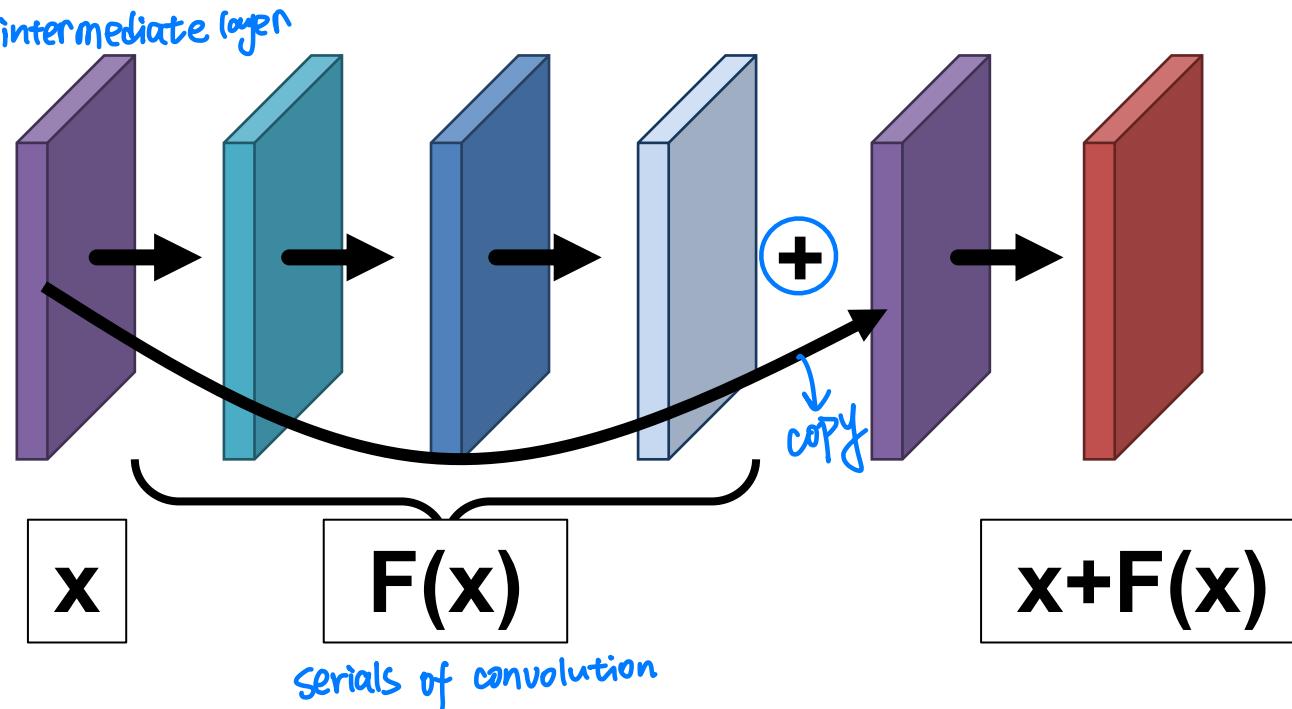


Residual Learning

New Building Block: $x + F(x)$

Mitigates vanishing/exploding gradients.

Lets you train networks with 100s of layers.



Evaluating Results

At training time, we minimize: $-\log\left(\frac{\exp((Wx)_{y_i})}{\sum_k \exp((Wx)_k)}\right)$

At test time, we evaluate, given predicted class \hat{y}_i :

Accuracy: $\frac{1}{n} \sum_{i=1}^n 1(y_i = \hat{y}_i)$

Evaluating Many Categories

Does this image depict a cat or a dog?



To avoid penalizing ambiguous images, many challenges let you make five guesses (top-5 accuracy):

Your prediction is correct if one of the guesses is right.

Accuracy over the Years

	Top 1 Error	Top 5 Error
Best Pre-Deep (~2012)	-	26.2%
Alexnet, 2012	43.5%	20.9%
VGG-16, 2014	28.4%	9.6%
ResNet-50, 2015	24.7%	7.8%
ResNet-152, 2015	21.7%	5.9%
ResNet-50 done better, 2018	20.7%	5.4%
Swin Transf., 2021	15.5%	-
ConvNeXt, 2022	14.5%	-
CoAtNet-7* 2021 (2B params!)	9.1%	-
Human*	-	5.1%

Many results from <https://paperswithcode.com/sota/image-classification-on-imagenet>. I am missing loads of great papers, and the numbers depend on tons of practical details. *Human – this number is from Andrej Karpathy and isn't really human performance with training but a ballpark. Resnet-50 one better = “Bag of Tricks for Image Classification with Convolutional Neural Networks”, He et al.

A Practical Aside

- People usually use hardware specialized for matrix multiplies (the card below does 13.4T flops if it's matrix multiplies).
- Convolution is a linear operator (matrix multiply!)
- A good answer to why we love homogeneous coordinates?
 - Makes rendering matrix multiplies →
 - leads to matrix multiplication hardware →
 - deep learning.



Training a CNN

- Download a big dataset
- Initialize network weights randomly
- for epoch in range(epochs):
 - Shuffle dataset
 - for each minibatch in datsaet.:
 - Put data on GPU
 - Compute gradient
 - Update gradient with SGD

Training a CNN from Scratch

Need to start w somewhere

- AlexNet: weights $\sim \text{Normal}(0, 0.01)$, bias = 1
- Popular “Xavier” and “Kaiming” initialization:
Uniform() and Normal() parameters are
dependent on the parameters of each layer.
E.g., the number of neurons

Take-home: important, but use default
implementations

Training a ConvNet

- Convnets typically have millions of parameters:
 - AlexNet: 62 million
 - VGG16: 138 million
 - ConvNeXt-L: 198M
- Convnets typically fit on ~1.2 million images
- Remember least squares: if we have fewer data points than parameters, we're in trouble
- Solution: need regularization / more data

Training a CNN – Weight Decay

SGD
Update

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \epsilon \frac{\partial L}{\partial \mathbf{w}_t}$$

+Weight
Decay

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \epsilon \mathbf{w}_t + \epsilon \frac{\partial L}{\partial \mathbf{w}_t}$$

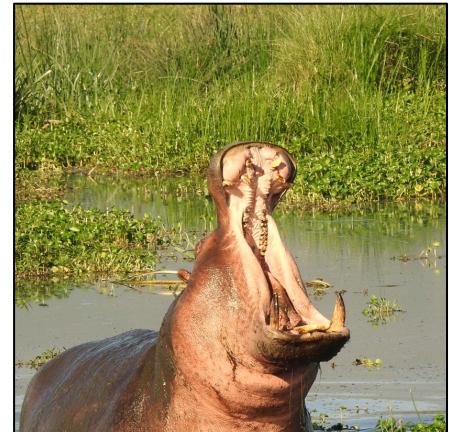
What does this remind you of?

Weight decay is similar to L2 weight regularization but is not be the same for more complex optimization techniques.

See “Decoupled Weight Decay Regularization”, Loshchilov and Hutter.

Quick Quiz

Raise your hand if it's a hippo



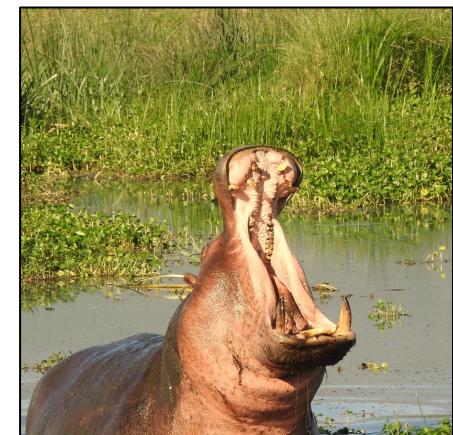
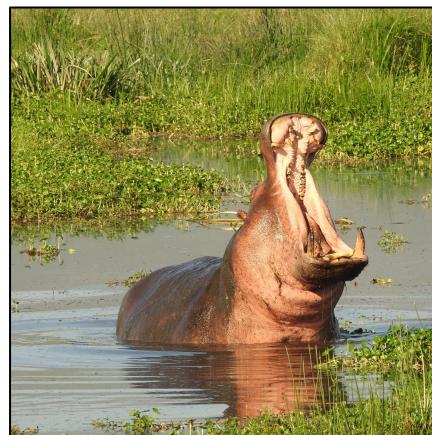
Horizontal
Flip

Color
Jitter

Image
Cropping

Training a CNN –Augmentation

- Apply transformations that don't affect the output
- Produces more data but you have to be careful that it doesn't change the meaning of the output

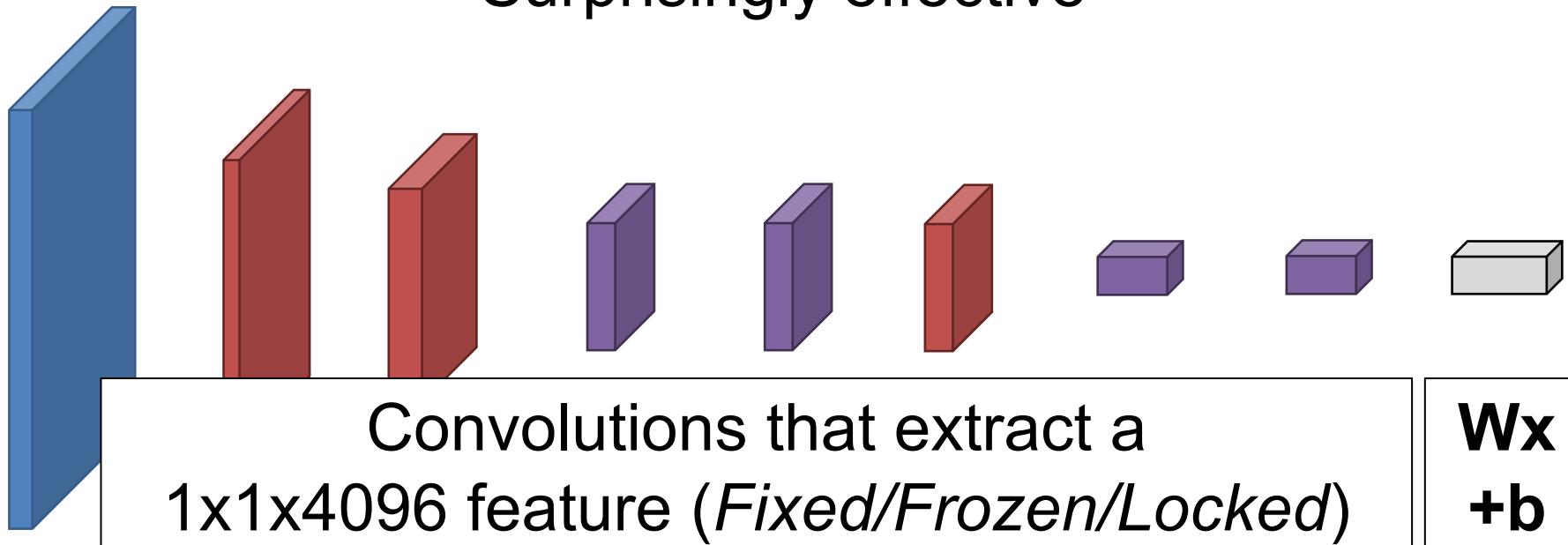


Training a CNN – Fine-tuning

- What if you don't have enough data?

Fine-Tuning: Pre-trained Features

1. Extract some layer from an existing network
 2. Use as your new feature.
 3. Learn a linear model.
- Surprisingly effective



Fine-Tuning: Transfer Learning

- Rather than initialize from random weights, initialize from some “pre-trained” model that does something else.
- Most common model is trained on ImageNet.
- Other pretraining tasks exist but are less popular/simple.

Fine-Tuning: Transfer Learning

Why should this work?

Transferring from objects (dog) to scenes (waterfall)



Recommendations

- <10K images: pre-trained features
- **Always** try fine-tuning
- >100K images: consider trying from scratch or transfer learning

Summary

- We learned about converting an image into a vector output (e.g., which of K classes is this image, or predict K continuous outputs)
- We learned about some building blocks for doing this

Extras if You're Curious

Classic Recognition

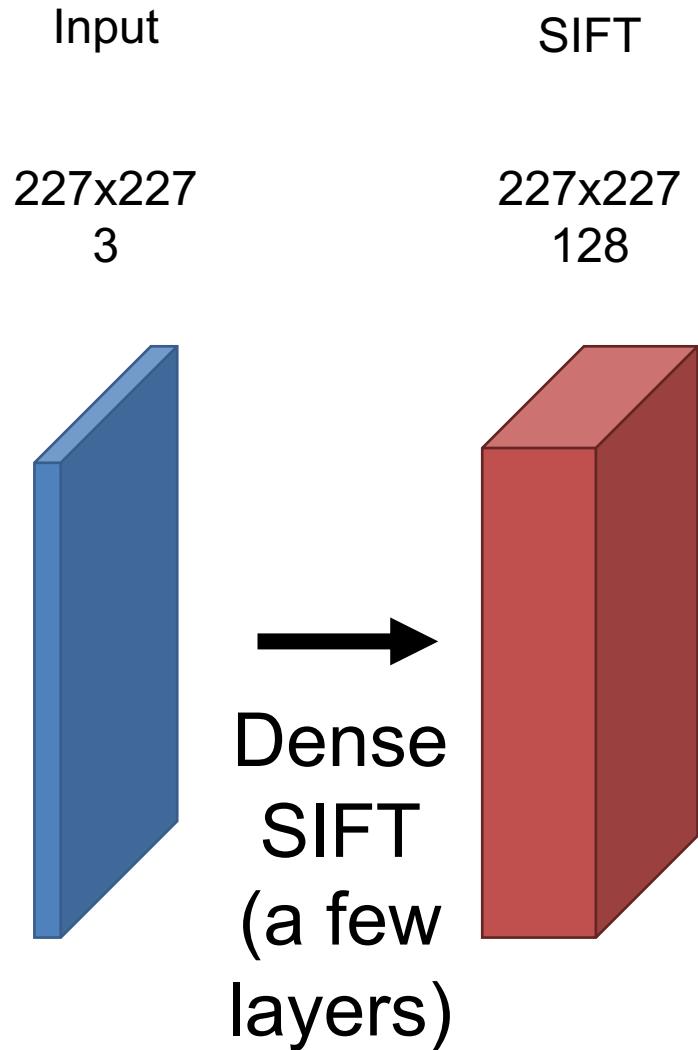
Input

227x227

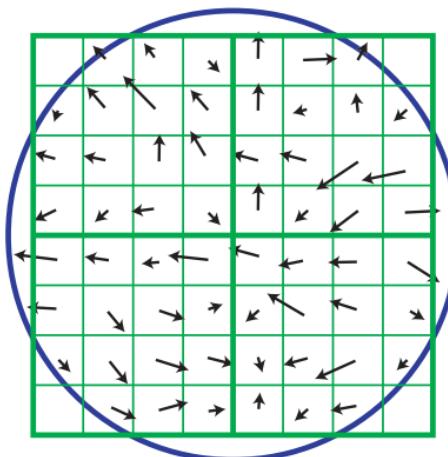
3



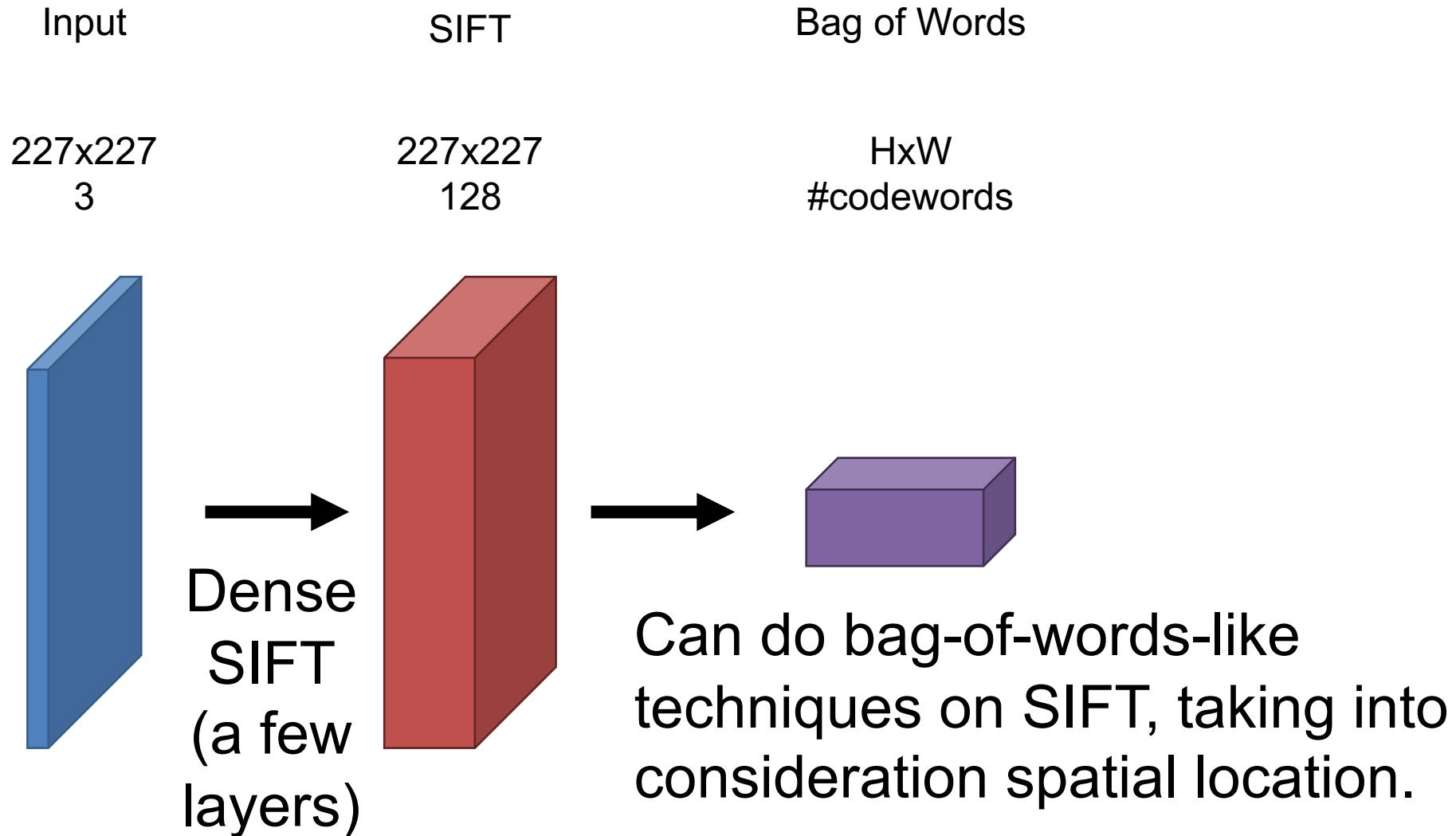
Classic Recognition



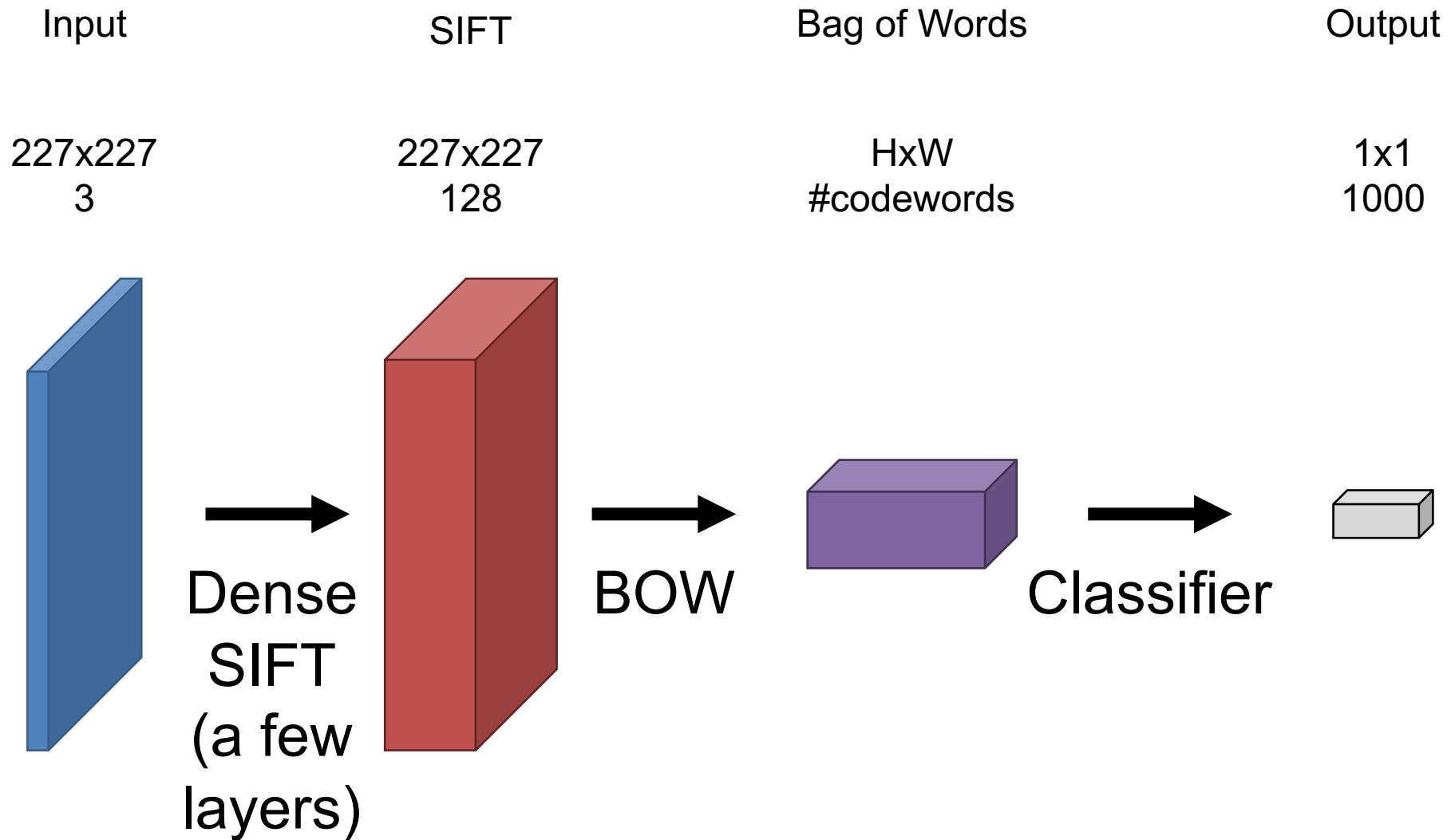
Recall: can compute a descriptor based on histograms of image gradients. Do it densely (at each pixel).



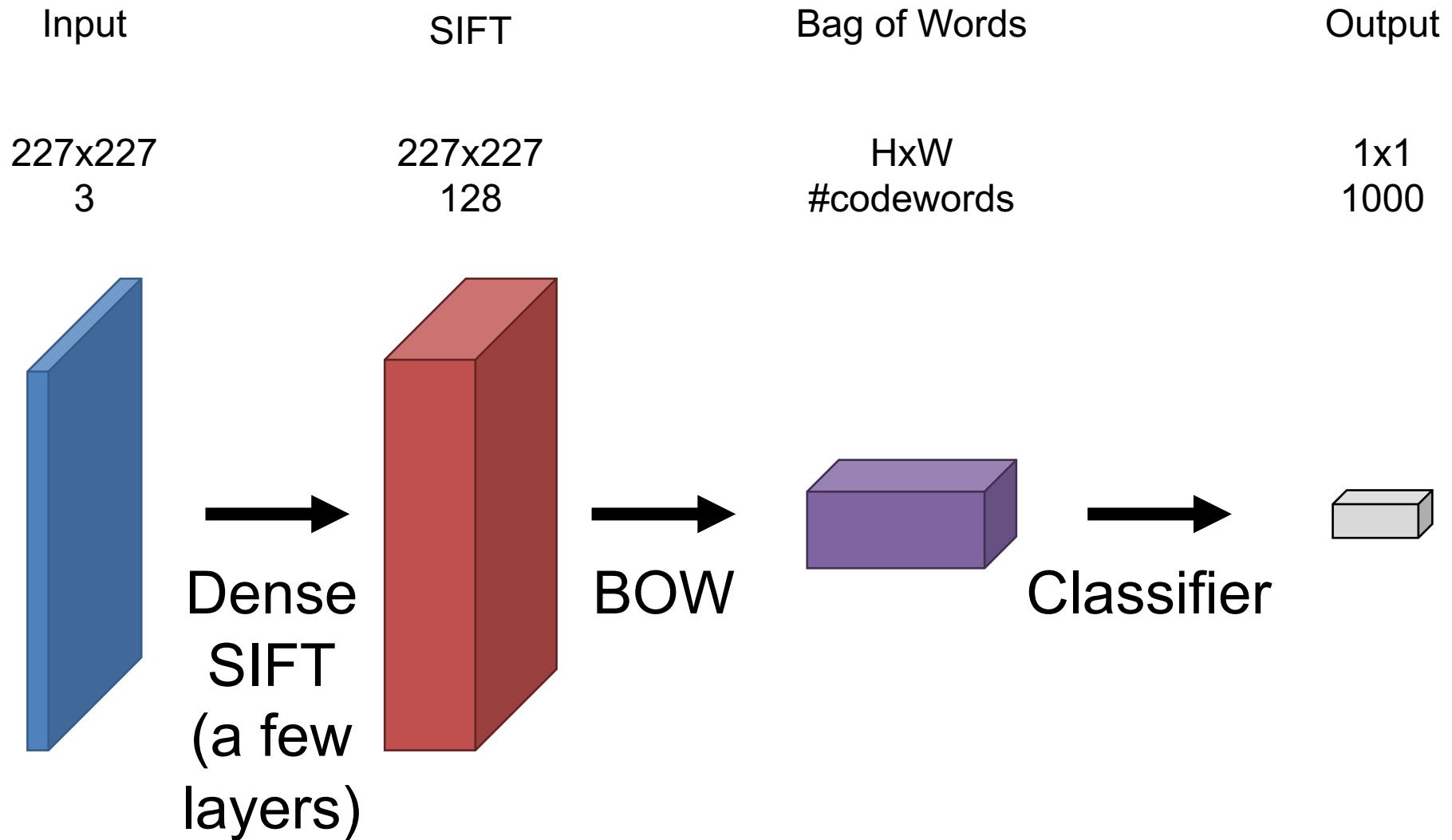
Classic Recognition



Classic Recognition



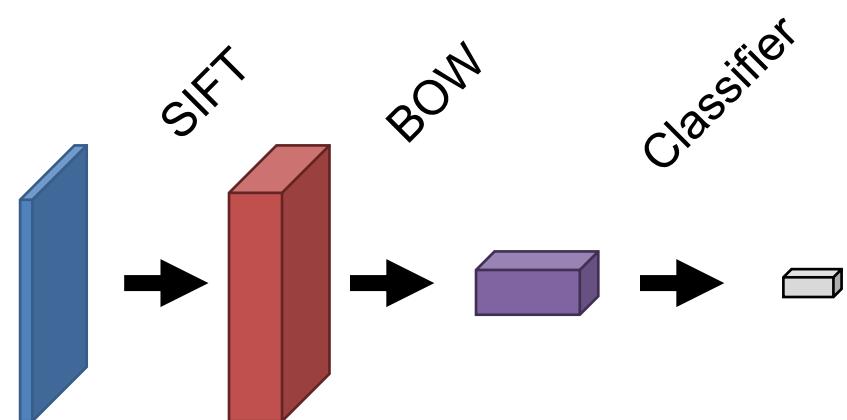
Classic Recognition



Classic vs Deep Recognition

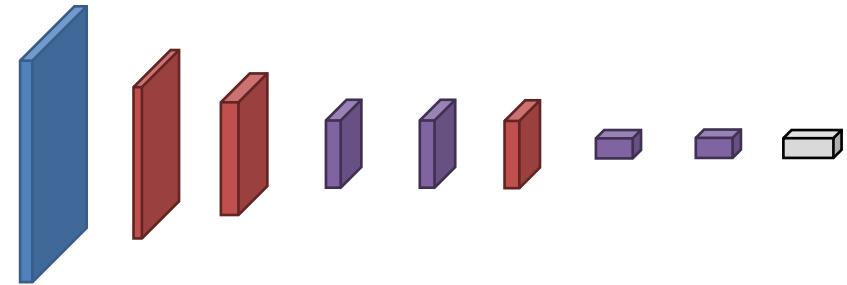
Classic

Pipeline of hand-engineered steps



Deep

Pipeline of learned convolutions + simple operations



What are some differences?

The classic steps don't talk to each other or have many parameters that are learned from data.